



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Desarrollo de Sistemas Distribuidos.

Tarea 9. Desarrollo de un prototipo de sistema de comercio electrónico utilizando un Serverless.

Nombre: Flores Castro Luis Antonio.

Profesor: Pineda Guerrero Carlos.

Grupo: 4CV13.

16 junio 2023.

Desarrollo.

Creación de proyecto en Visual Studio Community 2022.

Primero debemos abrir Visual Studio 2022, se desplegará una ventana y elegimos dentro del menú “Tareas Iniciales”, la opción “Crear un proyecto”.

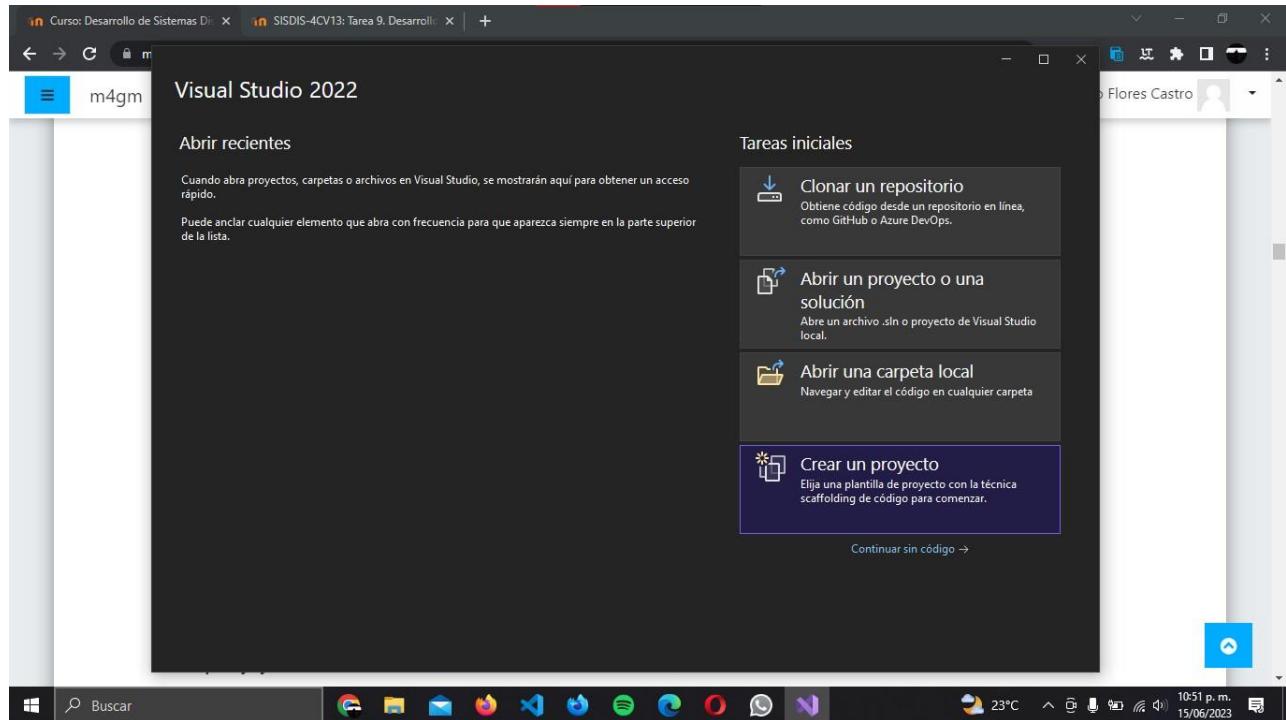


Figura 1. Creación de proyecto.

Se desplegará la siguiente pantalla, aquí buscaremos y elegiremos la opción “Azure Functions” y damos click en el botón siguiente.

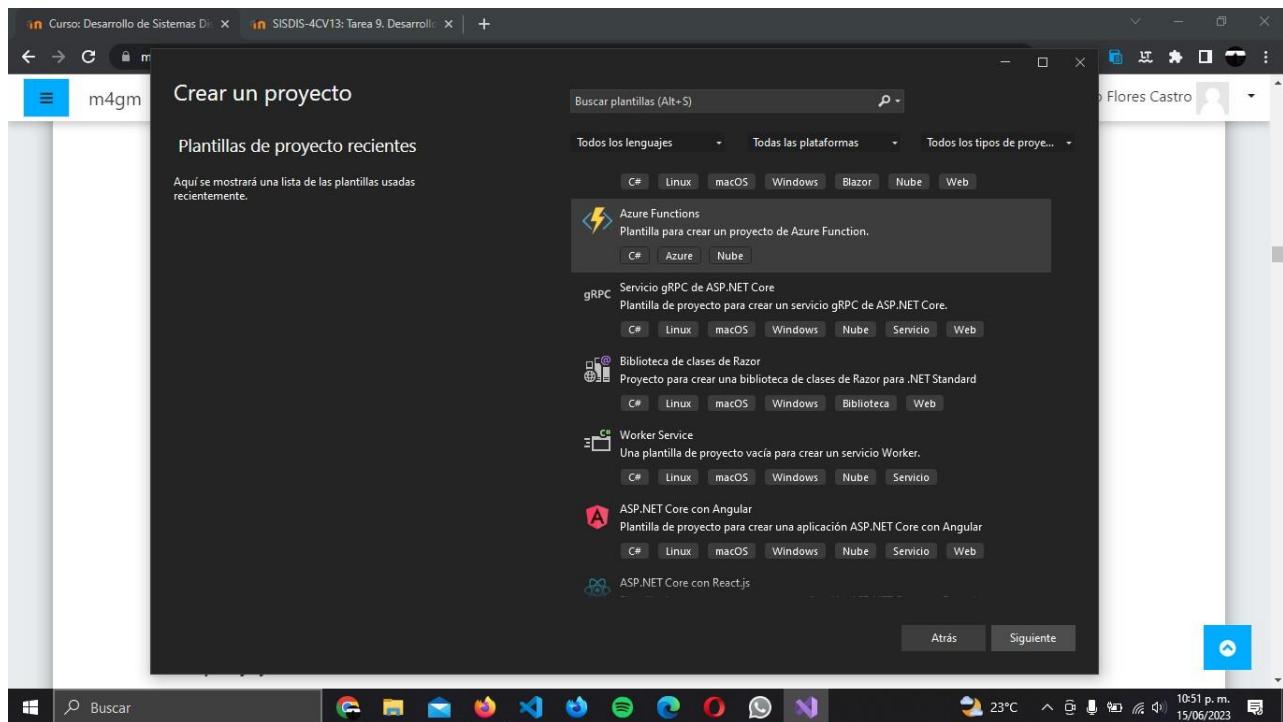


Figura 1.1. Opción “Azure Functions”.

Se desplegará una pantalla de configuración, se asigna un “*Nombre del proyecto*”, en este caso se dejó el nombre por default (FunctionApp1) y se activa la casilla “*Colocar la solución y el proyecto en el mismo directorio*” y damos click en el botón “*Siguiente*”.

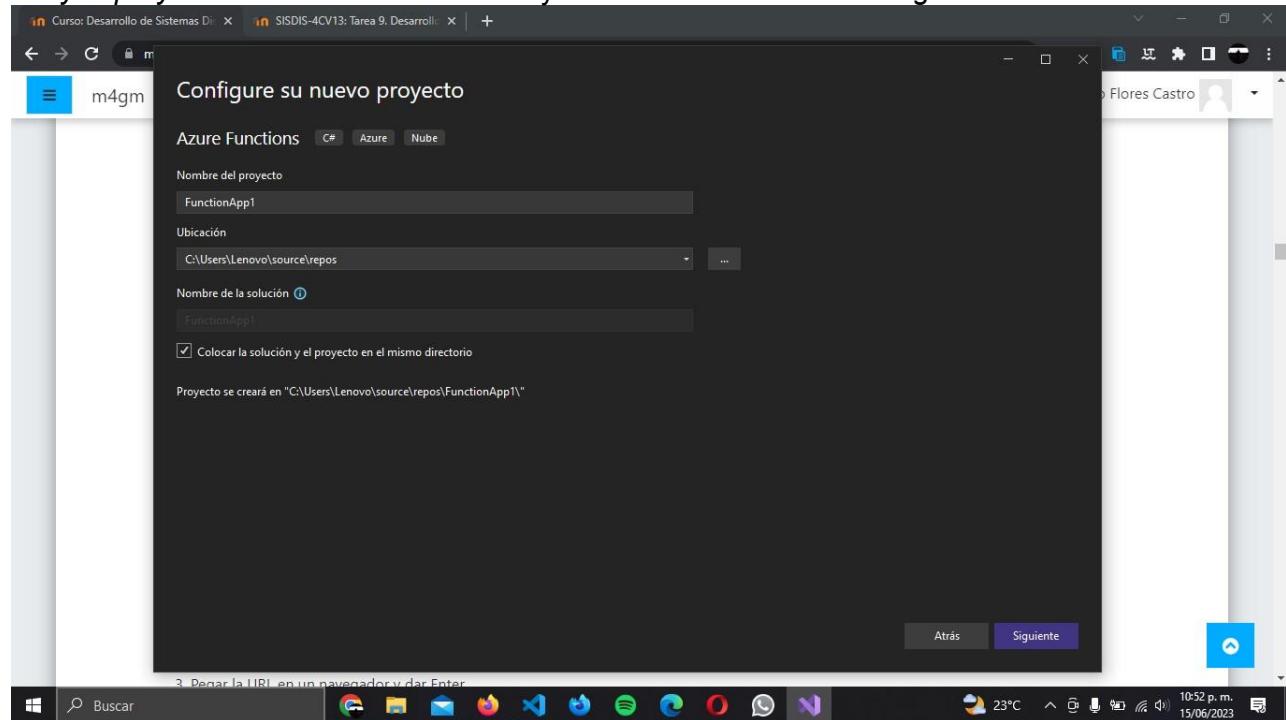


Figura 1.2. Configuración de proyecto.

Se desplegará una pantalla de “*Información adicional*”, aquí dejaremos por default la opción del campo “*worker de funciones*”, en el campo “*Función*”, elegiremos la opción HTTP trigger y finalmente activamos la casilla “*Usar Azurite....*” Y damos click en el botón “*Crear*”.

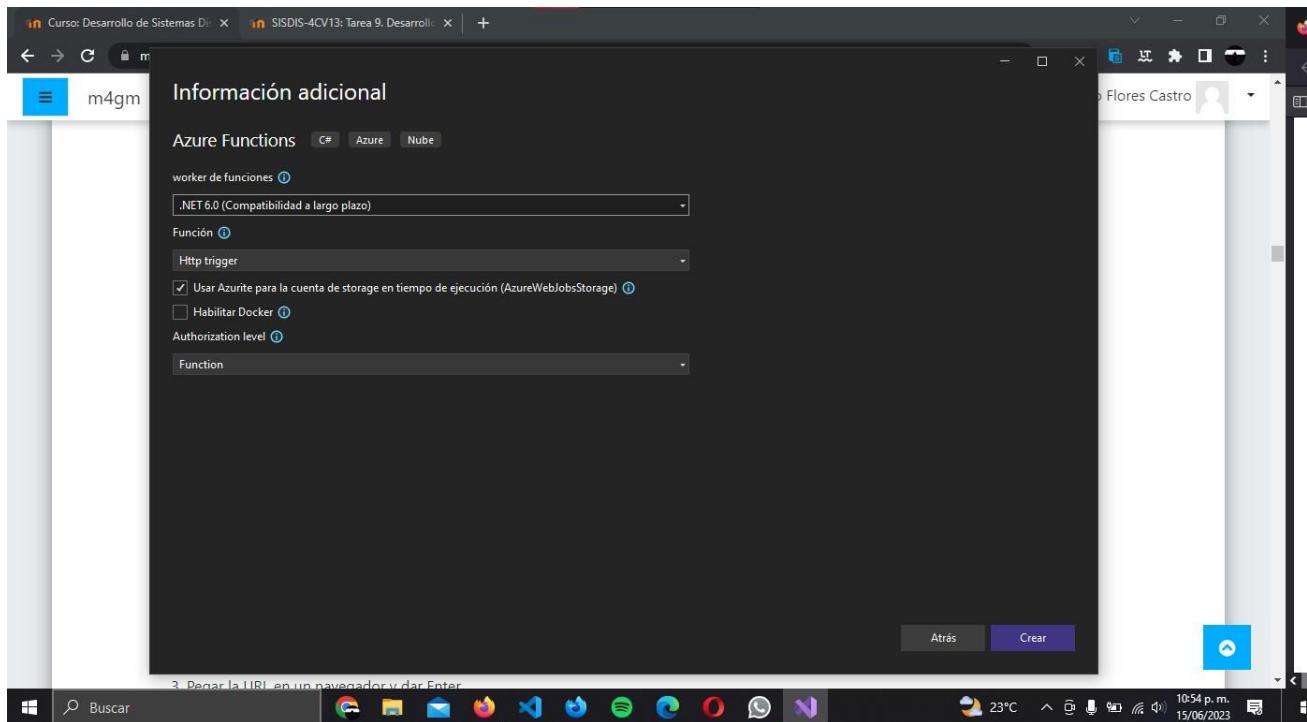


Figura 1.3. Información adicional del proyecto.

Podemos ver como se esta creando el proyecto.

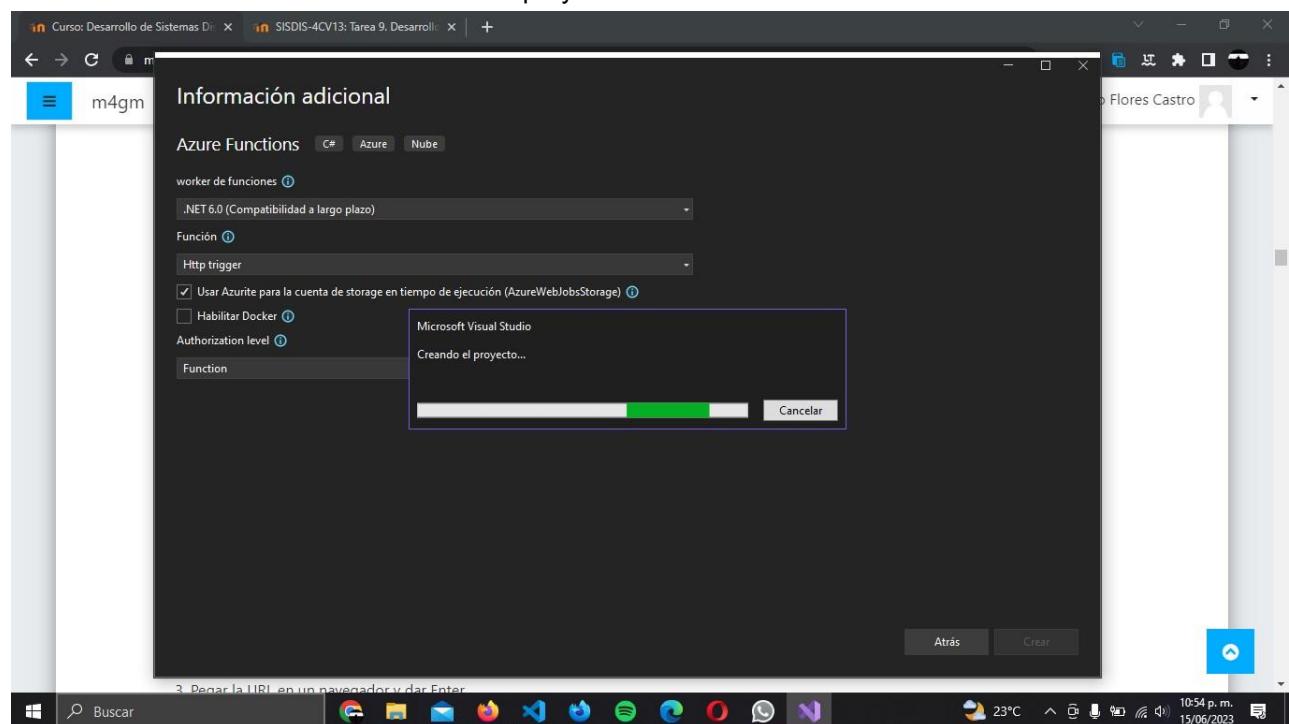


Figura 1.4. Creación de proyecto.

Una vez creado el proyecto, se abrirá la interfaz de Visual Studio 2022 con nuestro proyecto.

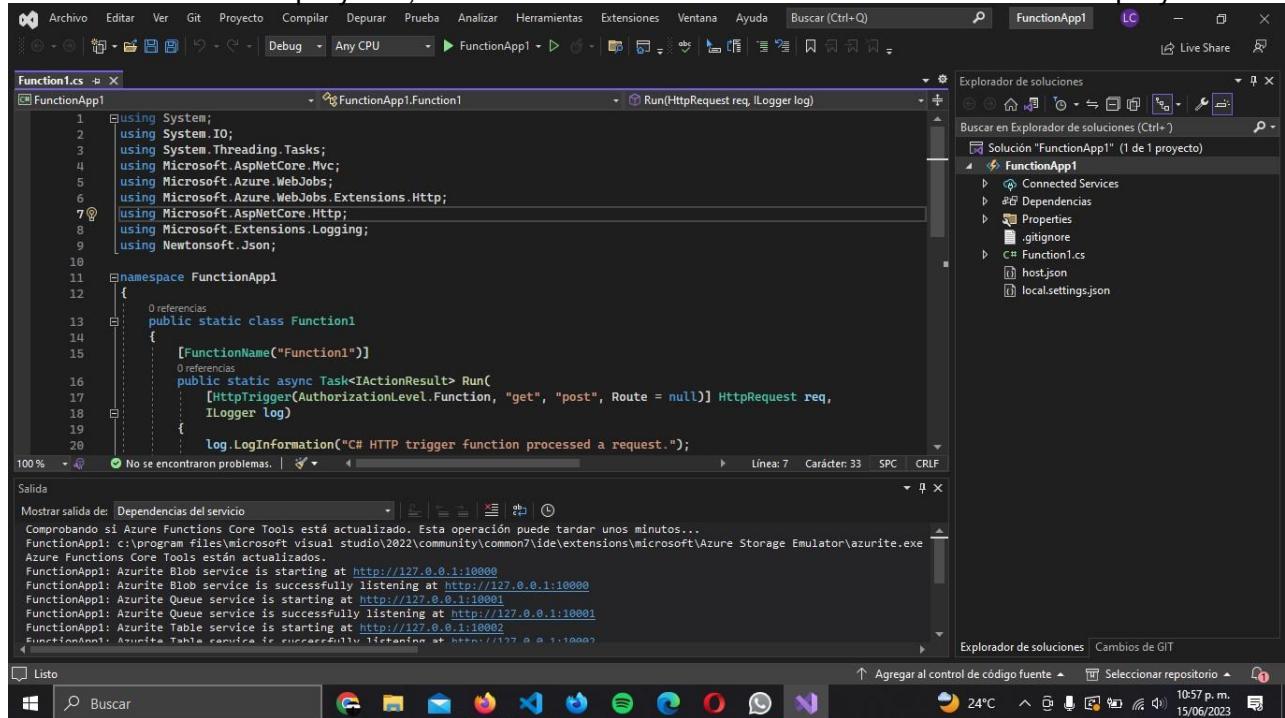


Figura 1.5. Despliegue de interfaz de Visual Studio 2022.

Ahora se realizará la publicación del proyecto en Azure, para debemos dar click derecho sobre el nombre de nuestro proyecto, este se encuentra en el “Explorador de soluciones”, en la parte derecha de la pantalla, después elegiremos la opción “Publicar”.

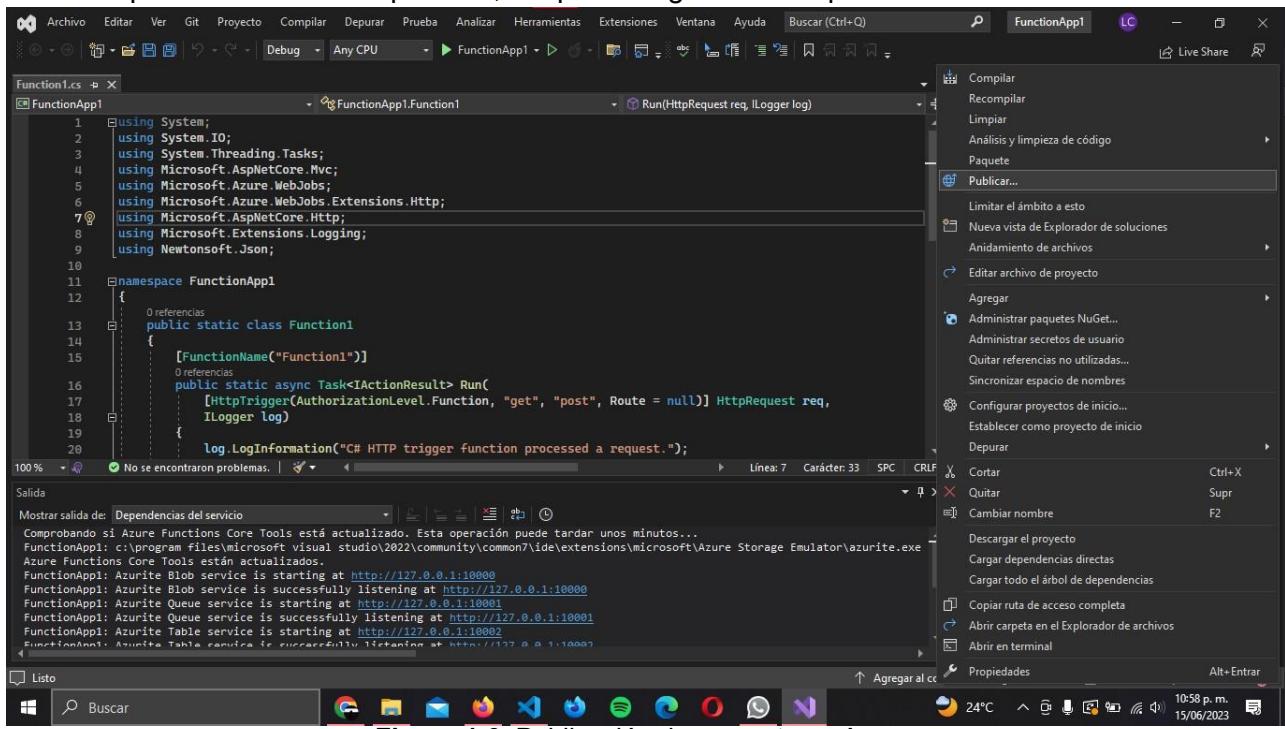


Figura 1.6. Publicación de proyecto en Azure.

Se desplegará la siguiente pantalla, aquí elegiremos la opción “Azure” y daremos click en “Siguiente.”

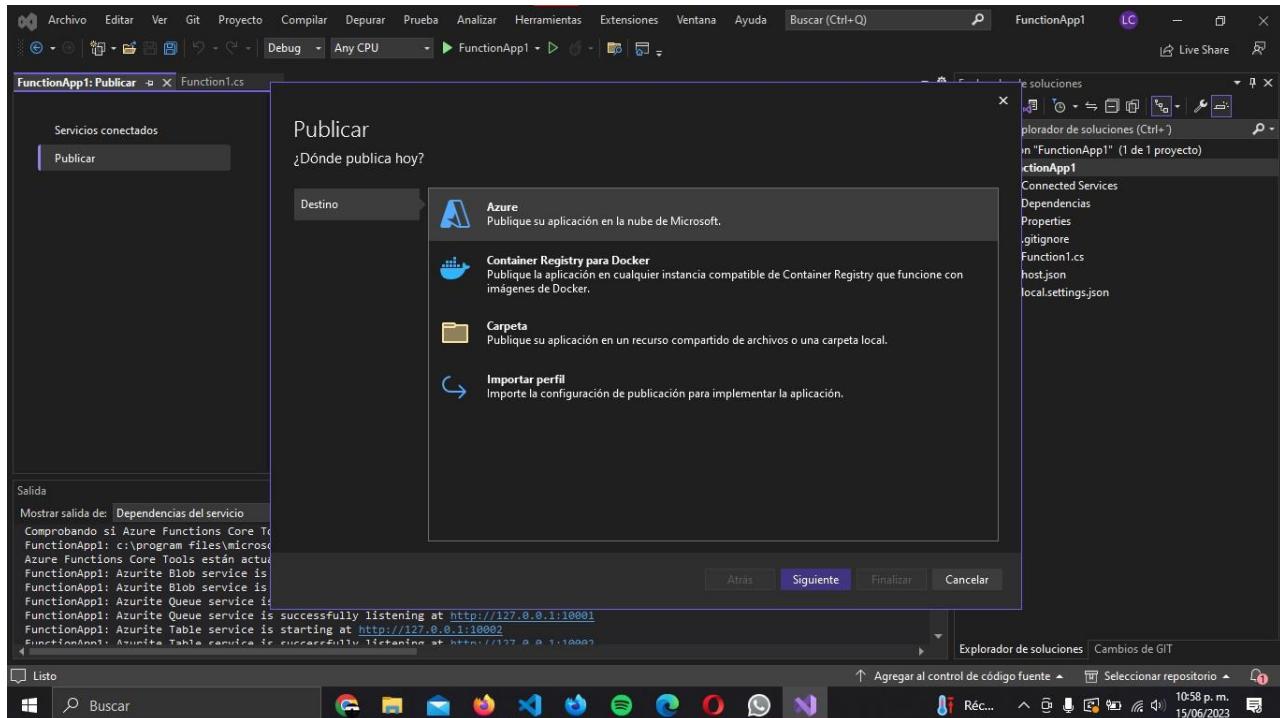


Figura 1.7. Publicación del proyecto en Azure.

En la siguiente pantalla, elegimos la opción “Azure Function App (Window)” y damos click en el botón “Siguiente”.

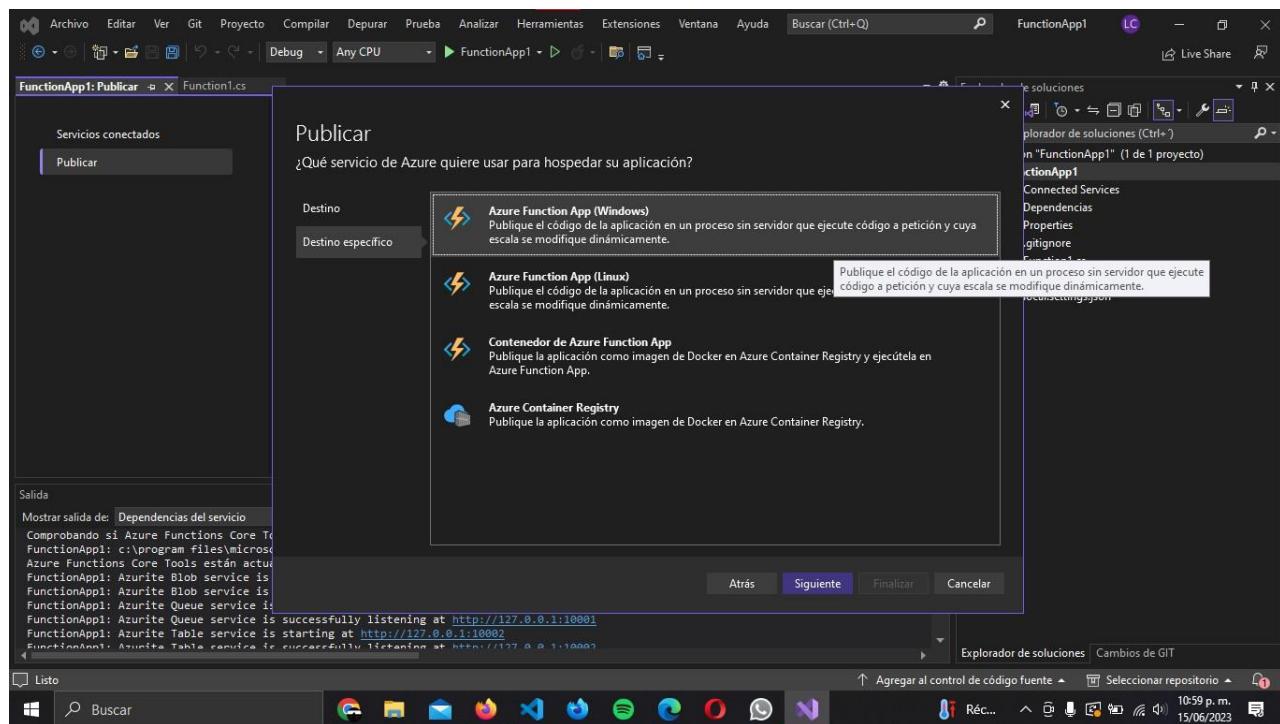


Figura 1.8. Publicación del proyecto en Azure.

Creación de aplicación de funciones.

Ahora crearemos una nueva aplicación de funciones, damos click en “+ Crear nuevo”.

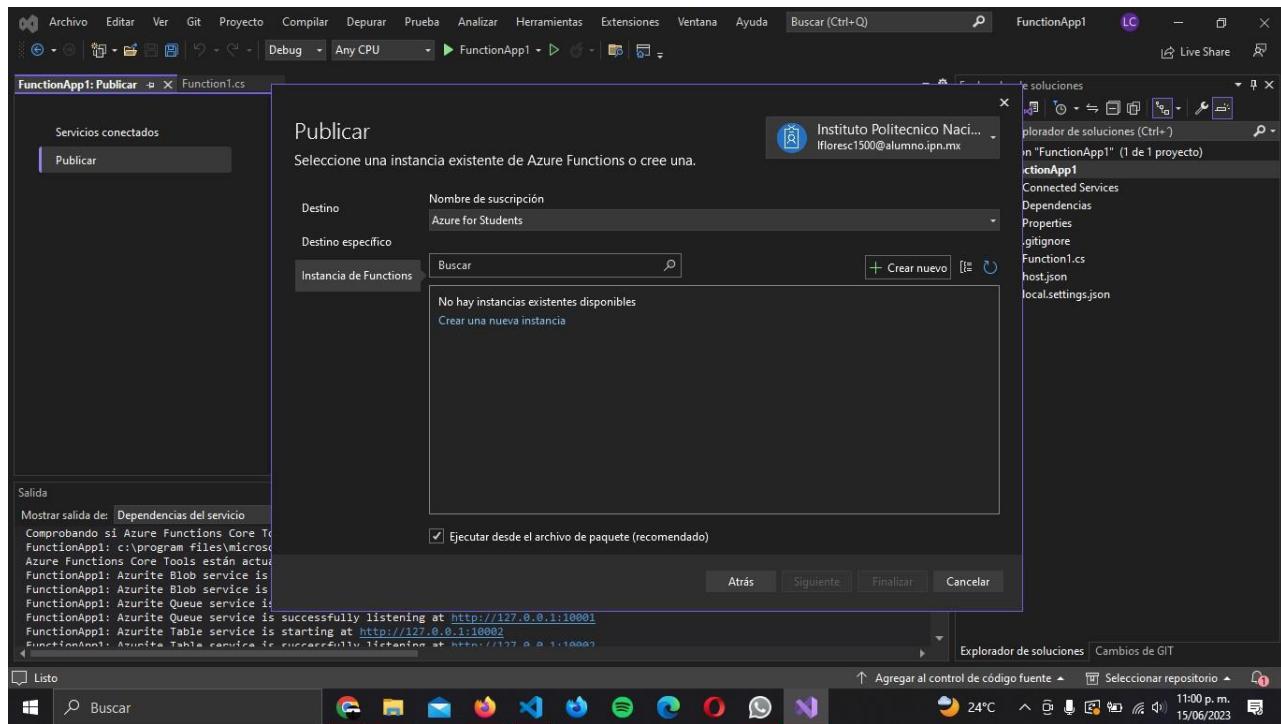


Figura 1.9. Creación de aplicación de funciones.

Se desplegará una pantalla e introduciremos los datos, el nombre que se le asignará a esta aplicación de funciones es “T9-AF-2019630463”, se elige un grupo de recursos, se selecciona una ubicación y en el campo “Azure Storage” damos click en la opción “Nuevo”.

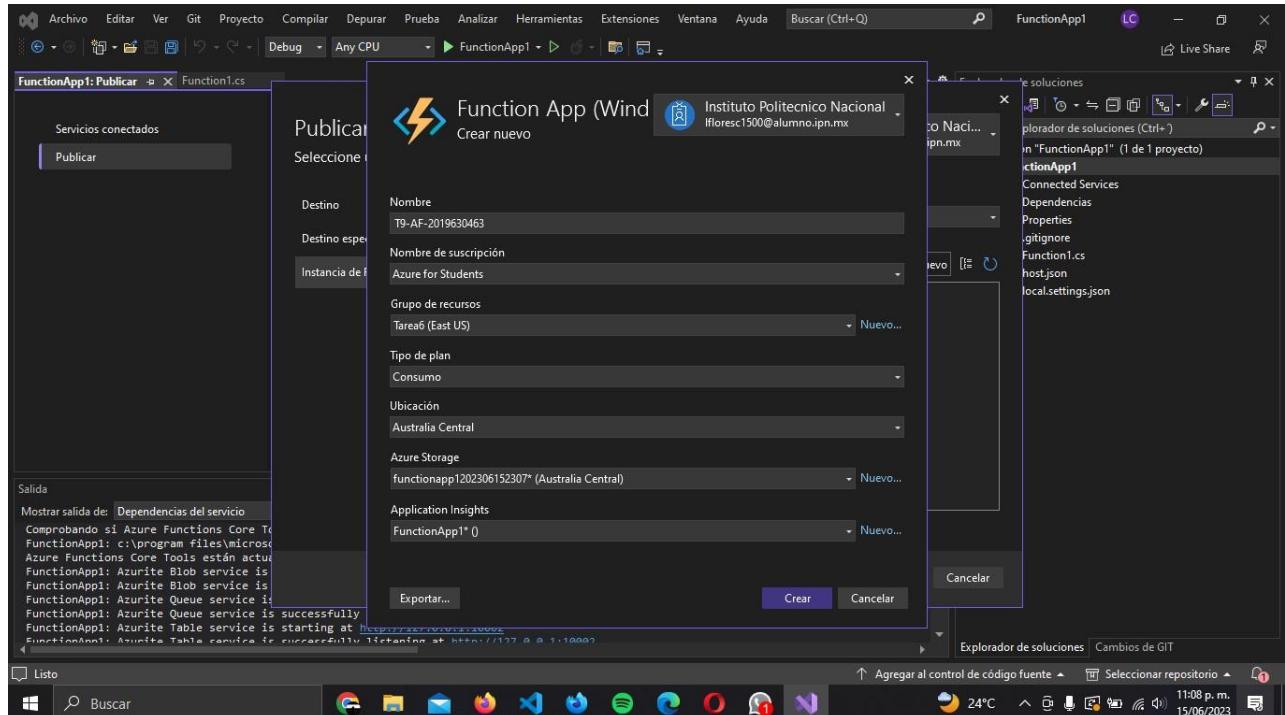


Figura 1.10. Creación de aplicación de funciones.

Se abrirá una ventana nueva, aquí solo daremos click en el botón “Aceptar”,

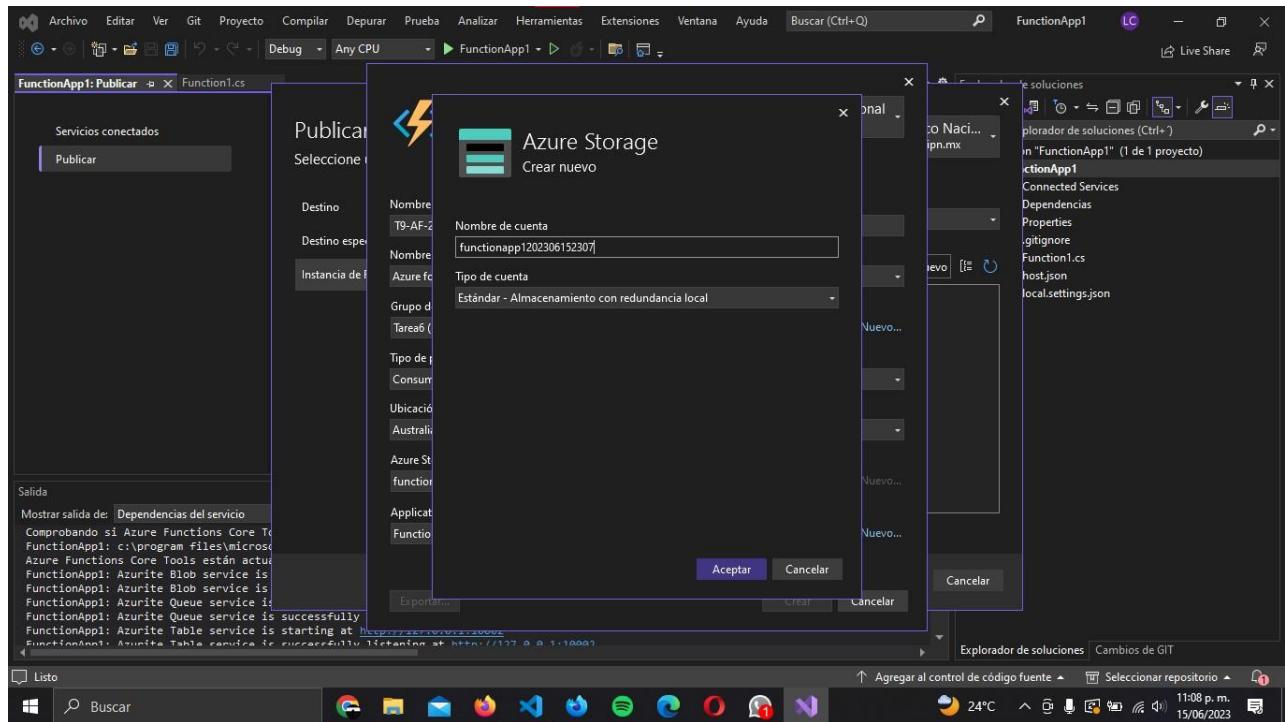


Figura 1.11. Creación de aplicación de funciones.

Finalmente, damos click en el botón “Crear”.

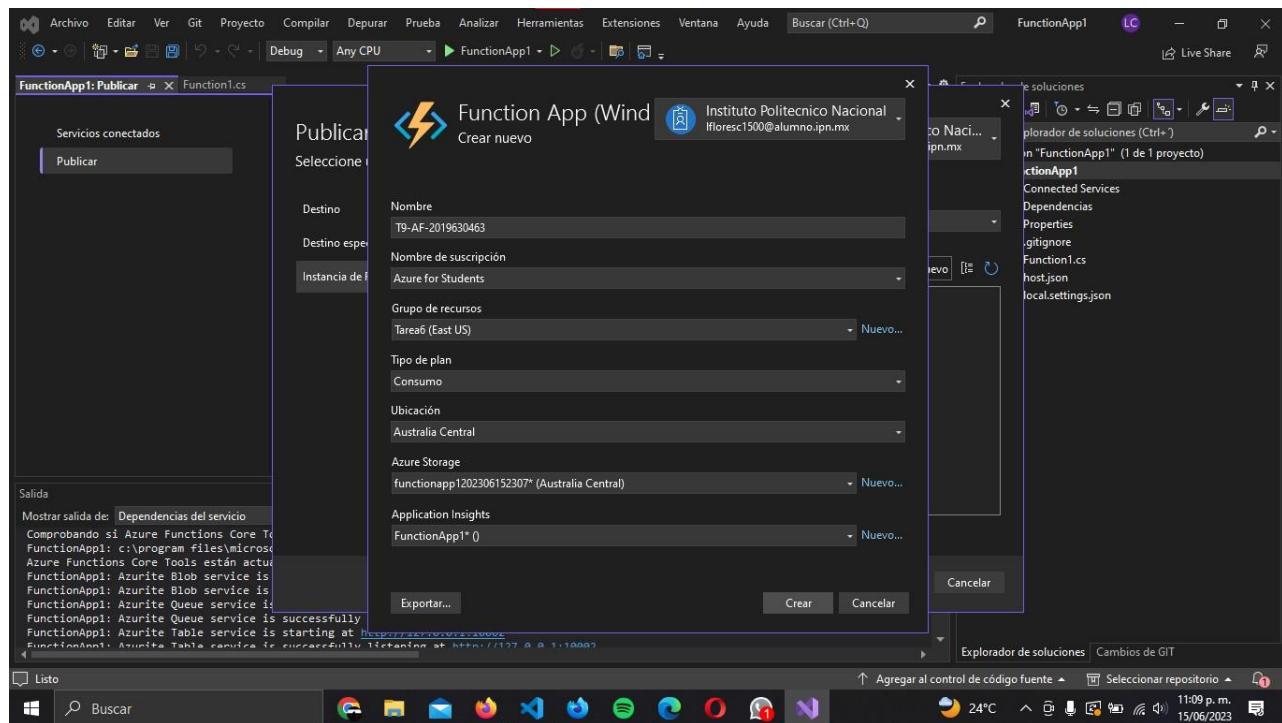


Figura 1.12. Creación de aplicación de funciones.

Se puede observar que se está creando.

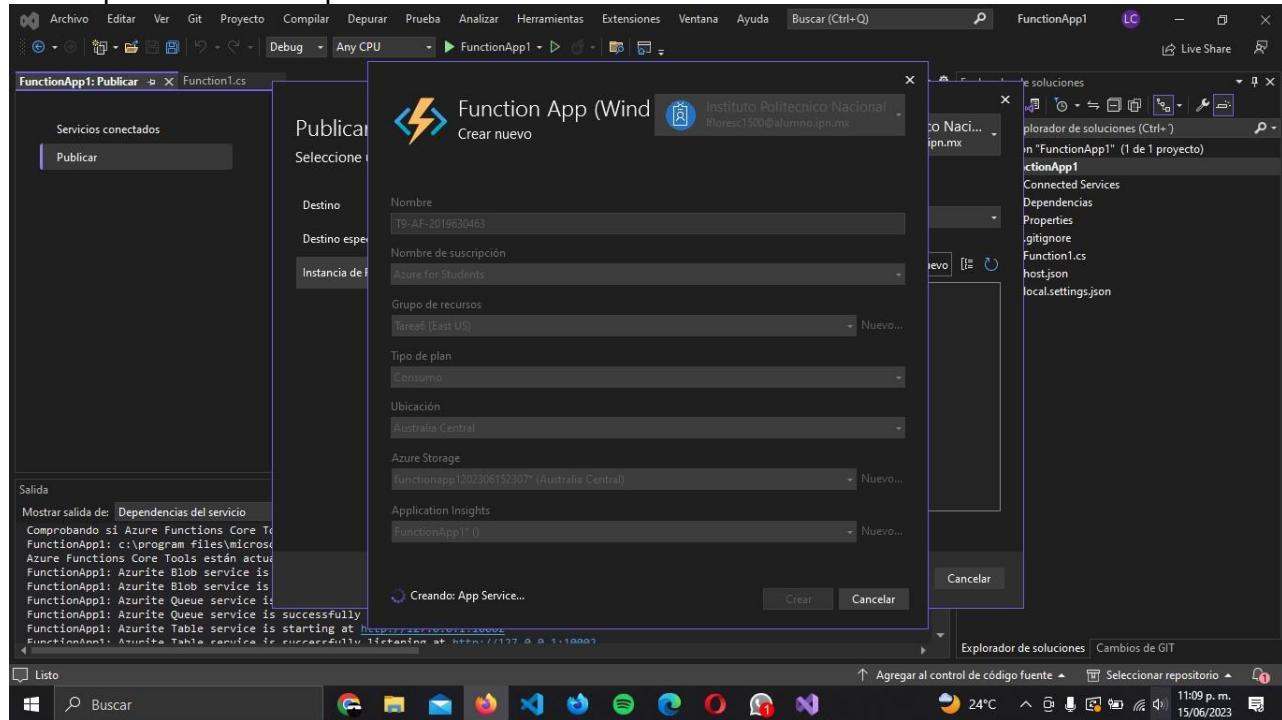


Figura 1.13. Creación de aplicación de funciones.

Se abre una pantalla y podemos ver que la aplicación de funciones “T9-AF-2019630463” ha sido creada, ahora damos click en el botón “Finalizar”.

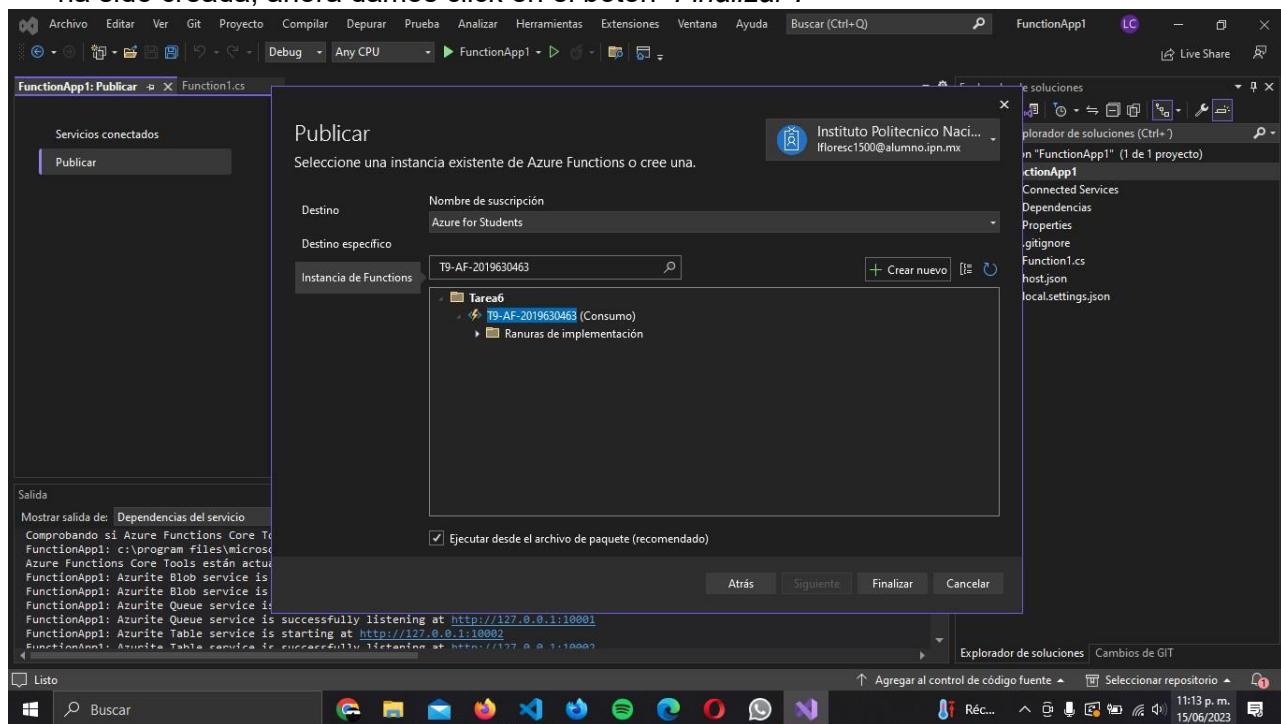


Figura 1.14. Aplicación de función creada.

Ahora en el portal de Azure, damos click en la opción “Aplicación de funciones”, esto para verificar que la aplicación de funciones creada ha sido publicada.

Nombre	Tipo	Última consulta
T6-2019630463-Imagen	Imagen	hace 3 semanas
vault505	Almacén de Recovery Services	hace 3 semanas
Tarea6	Grupo de recursos	hace 3 semanas

Figura 1.15. Aplicación de funciones en Azure.

Podemos ver que la aplicación de funciones creada ya se publicó en Azure, ahora seleccionamos la aplicación de funciones.

The screenshot shows the Microsoft Azure portal interface. At the top, there are three tabs: 'Curso: Desarrollo de Sistemas' (active), 'SISDIS-4CV13: Tarea 9. Desarroll...', and 'Aplicación de funciones - Microsoft'. The main content area is titled 'Aplicación de funciones' and shows a single item: 'T9-AF-2019630463'. Below the title, there are columns for 'Nombre' (T9-AF-2019630463), 'Estado' (En ejecución), 'Ubicación' (Australia Central), 'Plan de tarifa' (Dinámico), 'Plan de App Service' (AustraliaCentralPlan), 'Suscripción' (Azure for Students), and 'Tipo de...' (Function App). At the bottom, there are navigation buttons for 'Anterior', 'Página 1 de 1', and 'Siguiente >'. The status bar at the bottom right shows the date and time: '15/06/2023 11:15 p. m.'.

Figura 1.16. Aplicación de funciones en Azure.

Dentro de la aplicación de funciones, damos click en el menú “*Funciones*” y damos click en la opción “+ Crear”. Esto para crear una función dentro de la aplicación.

The screenshot shows the Microsoft Azure portal interface, specifically the 'T9-AF-2019630463 | Funciones' blade. On the left, there is a sidebar with sections: 'Introducción', 'Control de acceso (IAM)', 'Etiquetas', 'Diagnosticar y solucionar problemas', 'Microsoft Defender for Cloud', and 'Eventos (versión preliminar)'. The 'Funciones' section is expanded, showing 'Claves de la aplicación', 'Archivos de la aplicación', and 'Servidores proxy'. Below this is the 'Implementación' section with 'Espacios de implementación'. In the center, there is a search bar 'Buscar' and a 'Crear' button. A message 'No hay ningún resultado.' is displayed. At the bottom, there are navigation buttons for 'Página 1 de 1' and the Windows taskbar with various icons.

Figura 1.17. Conexión a máquina virtual mediante ssh.

Se desplegará la pantalla “Crear función” dentro del menu “Plantilla” elegiremos la opción “HTTP trigger” y damos click en la opción “Crear”.

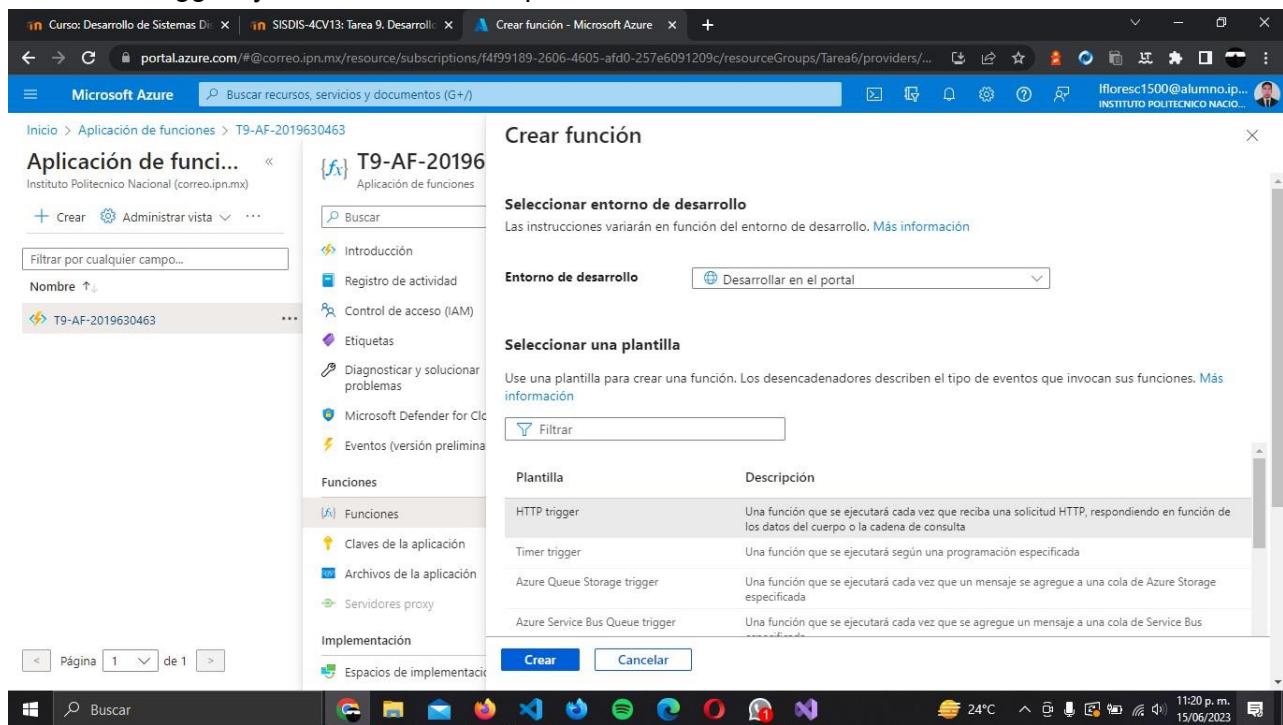


Figura 1.18. Creación de una función.

Después de algunos instantes, podemos ver que la función ha sido creada exitosamente.

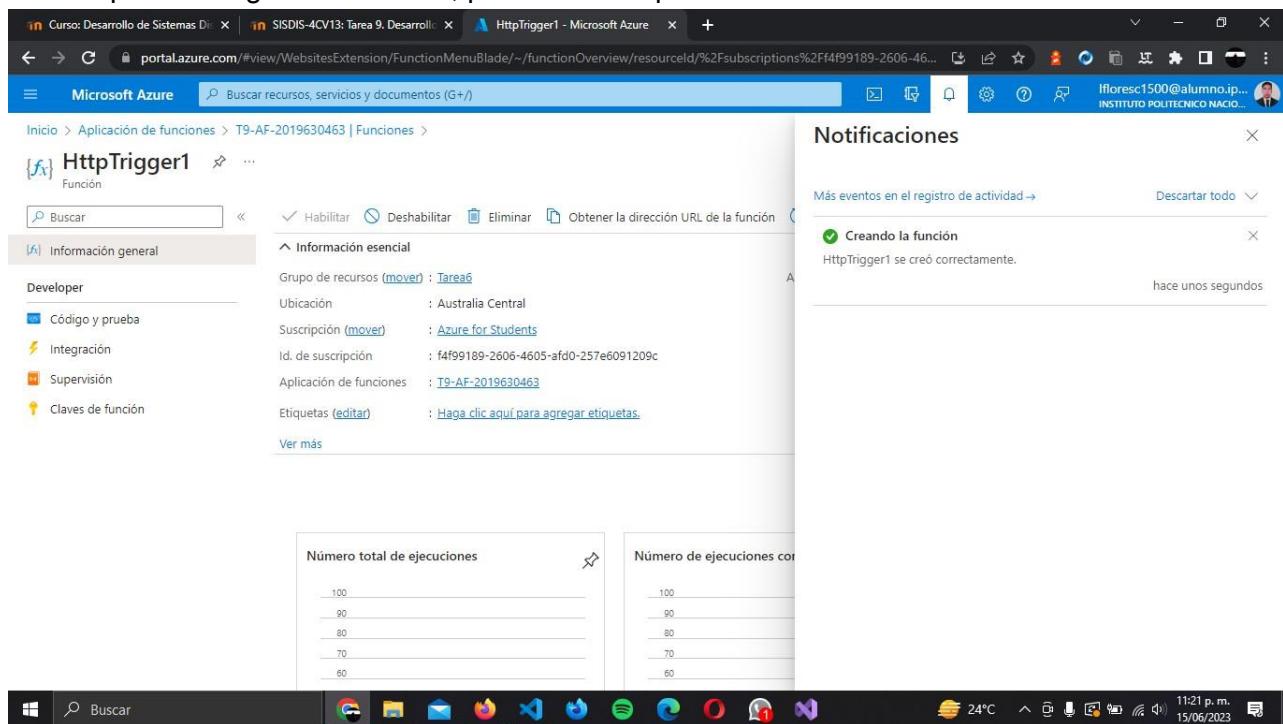


Figura 1.19. Función creada.

Ahora damos click en la opción “Obtener la dirección de la función”.

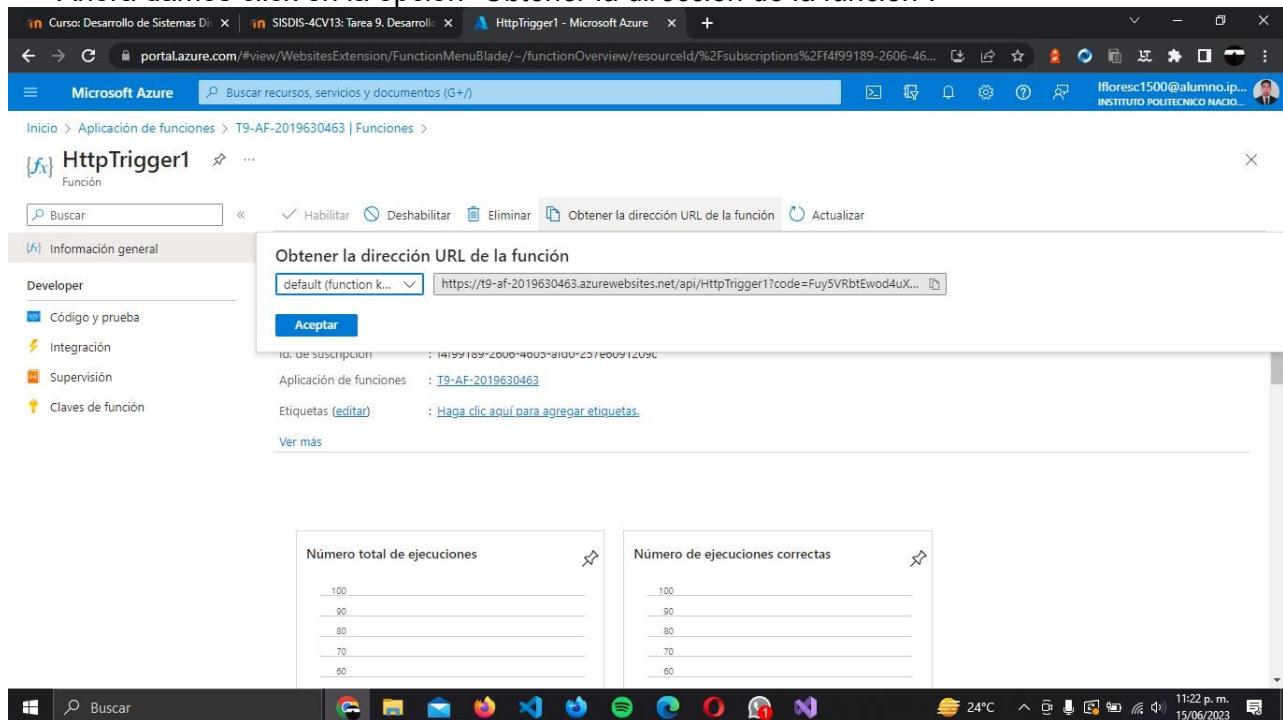


Figura 1.20. Transferencia de archivos del servicio web a máquina virtual.

Finalmente, para ver que la aplicación de funciones y la función creada funcionen de manera adecuada, copiamos la URL obtenida, al final de esta URL se puso “&name=Luis”, de esta forma se mostrará un mensaje.

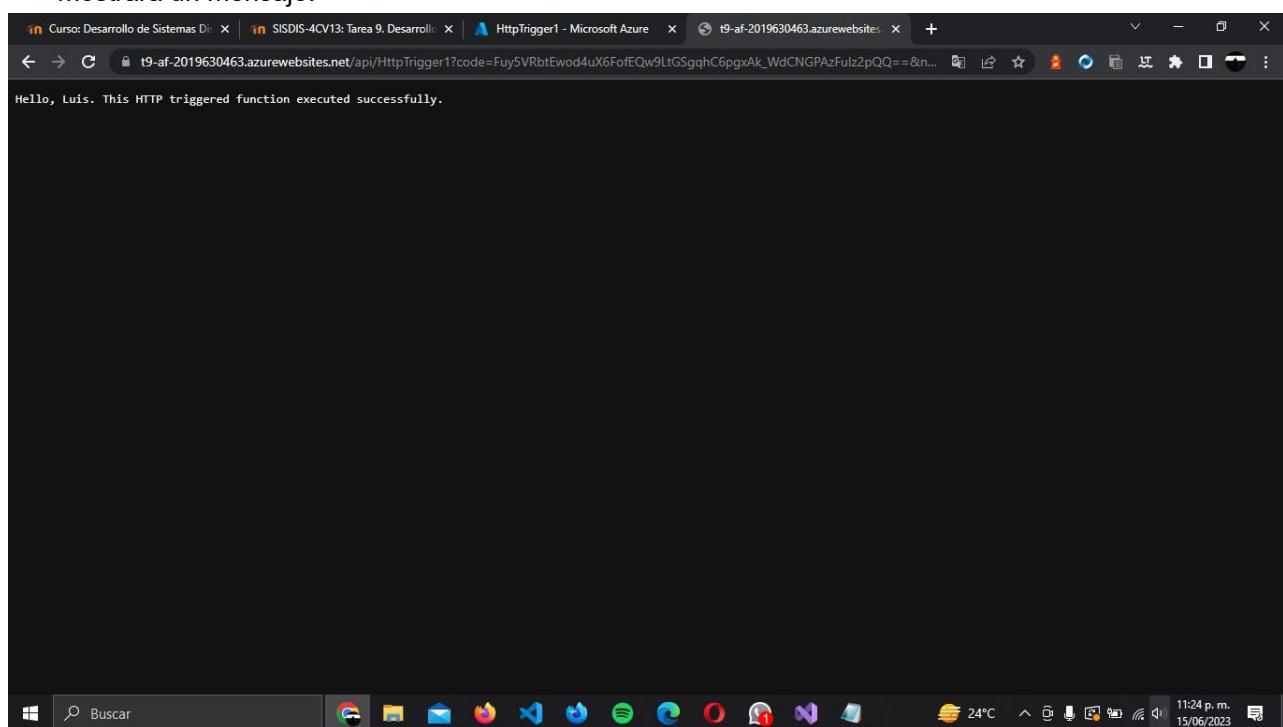
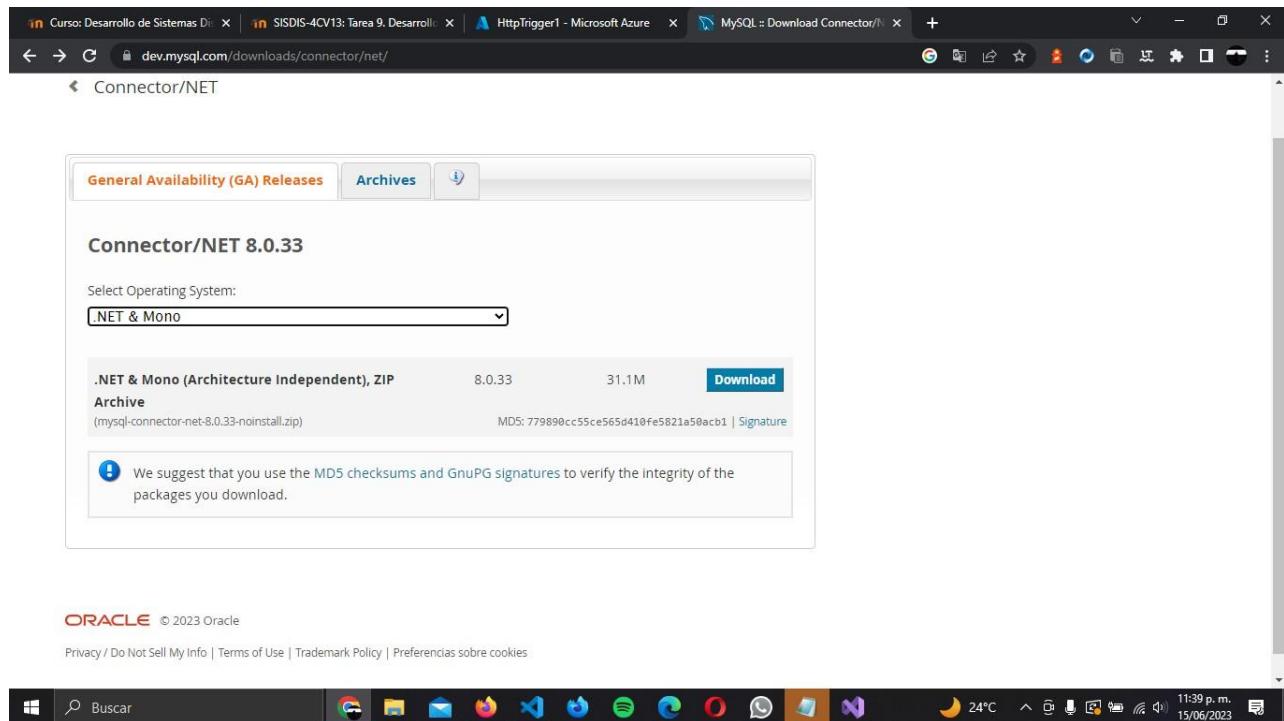


Figura 1.21. Funcionamiento correcto de función.

Instalar MYSQL Connector/.NET en Visual Studio.

Primero descargaremos el connector del URL: <https://dev.mysql.com/downloads/connector/net>. Una vez dentro, en el campo “Select Operating System” la opción “.NET & MONO” y damos click en “Download”.



1.22. Instalación MYSQL connector.

Se redireccionará a otra pagina donde daremos click en la opción “*No thanks, just start my download*” y el archivo .zip se comenzará a descargar.

Login Now or Sign Up for a free account.
An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account **Sign Up »**
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.



Figura 1.23. Descarga de MYSQL Connector.

Ya descargado, descomprimimos el archivo .zip.

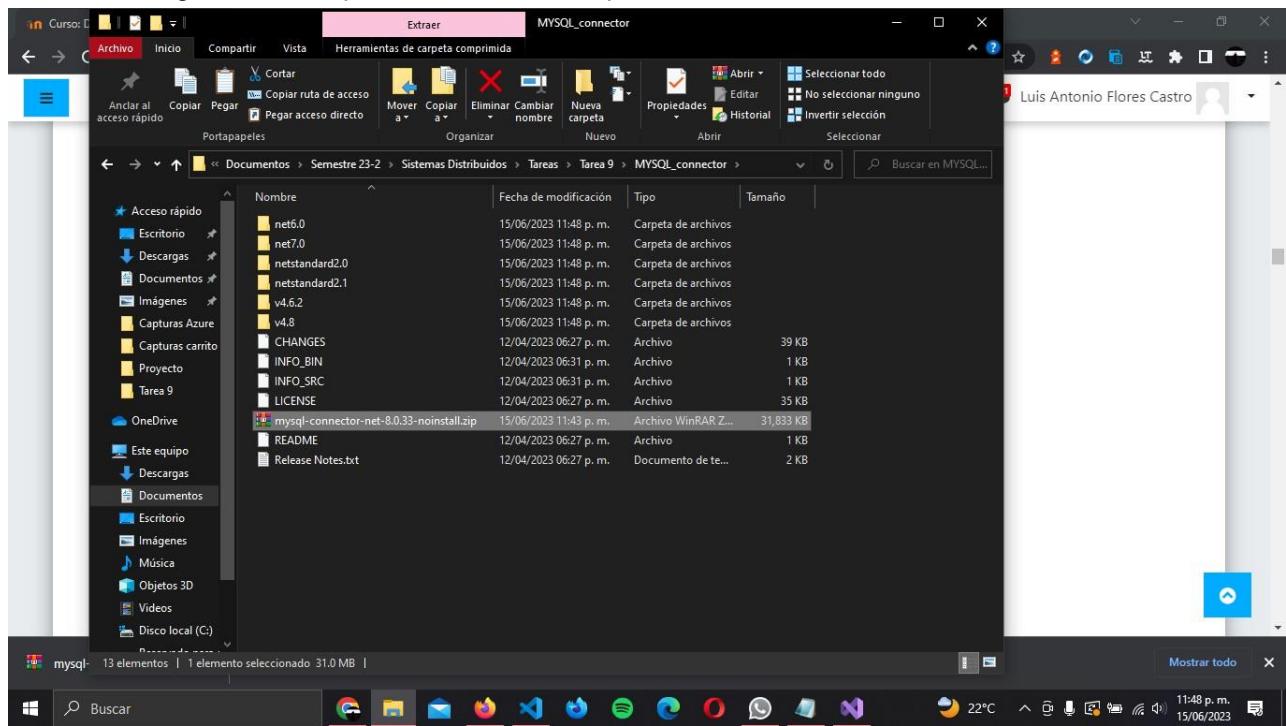


Figura 1.24. Descomprimir el arhcivo .zip MYSQL Connector.

Dentro de Visual Studio 2022, en la pestaña “Proyecto” damos click y ahora elegimos la opción “Agregar referencia del proyecto...”

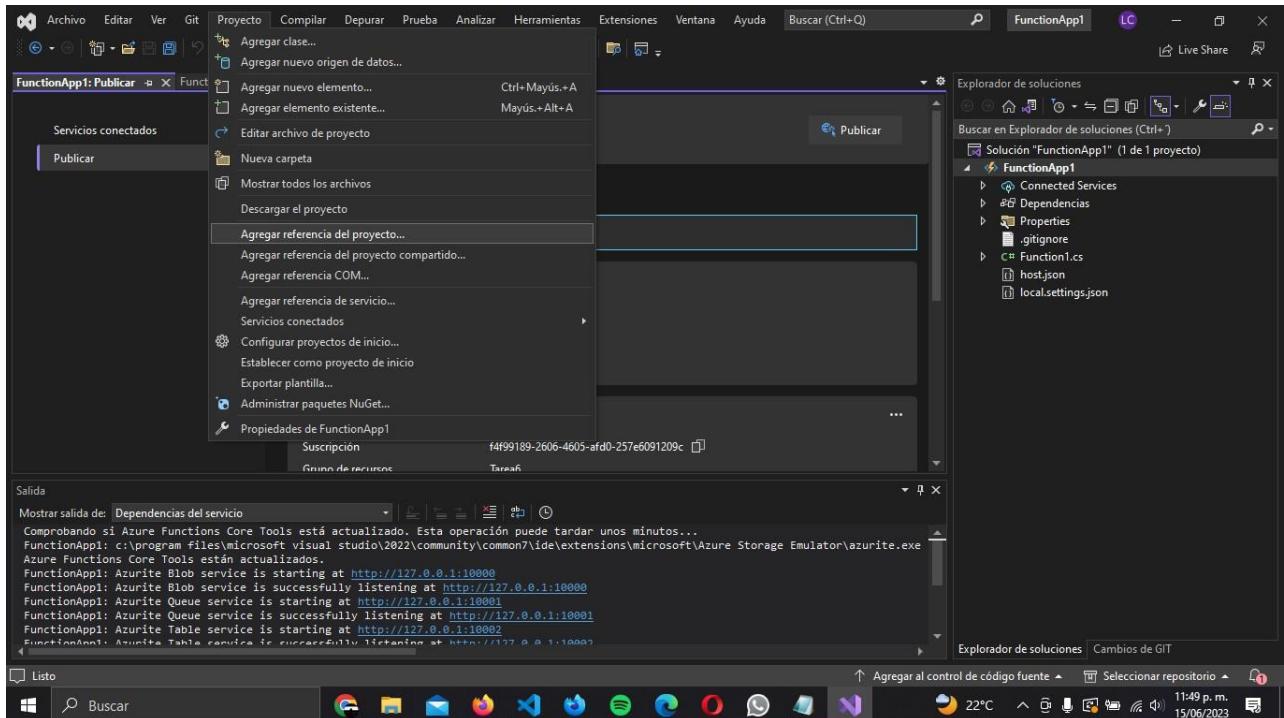


Figura 1.25. Instalación MySQL Connector.

Se mostrará la siguiente pantalla y daremos click en el botón “Examinar”.

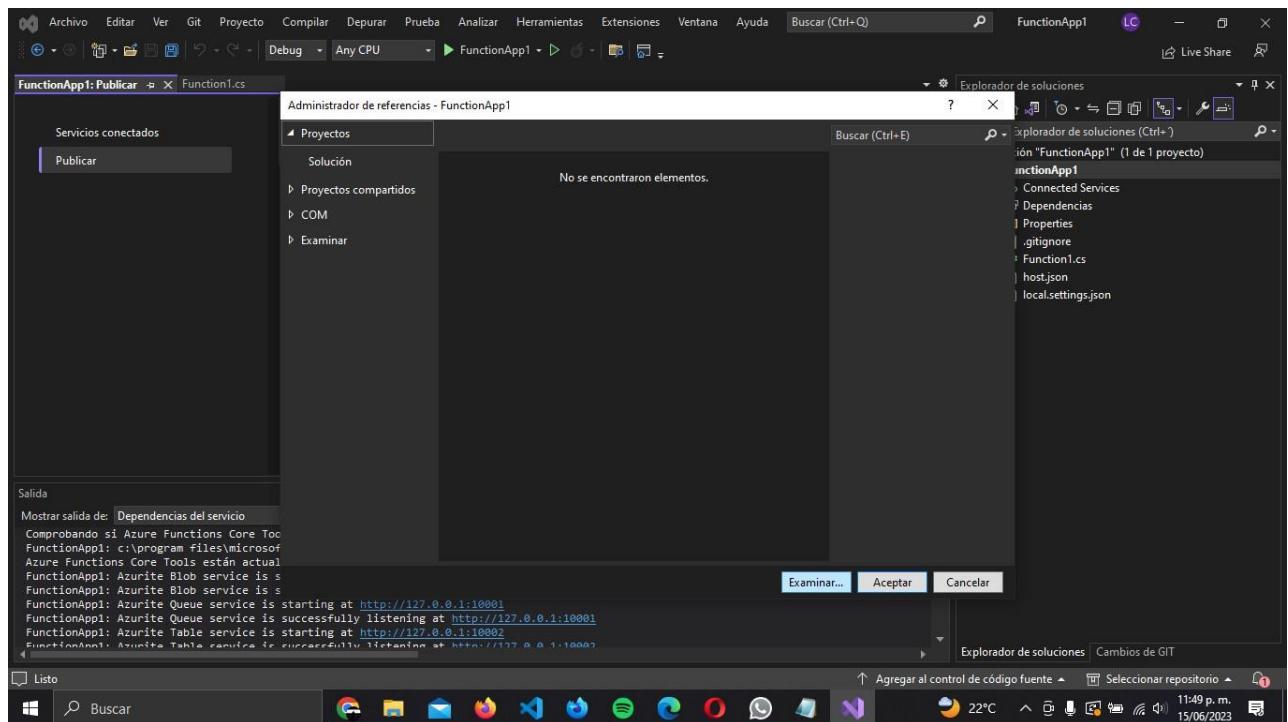


Figura 1.26. Instalación MySQL Connector.

Del archivo .zip que se descomprimió elegiremos la carpeta “net7.0” y seleccionamos el archivo “MySQL.Data.dll” y damos click en el botón agregar.

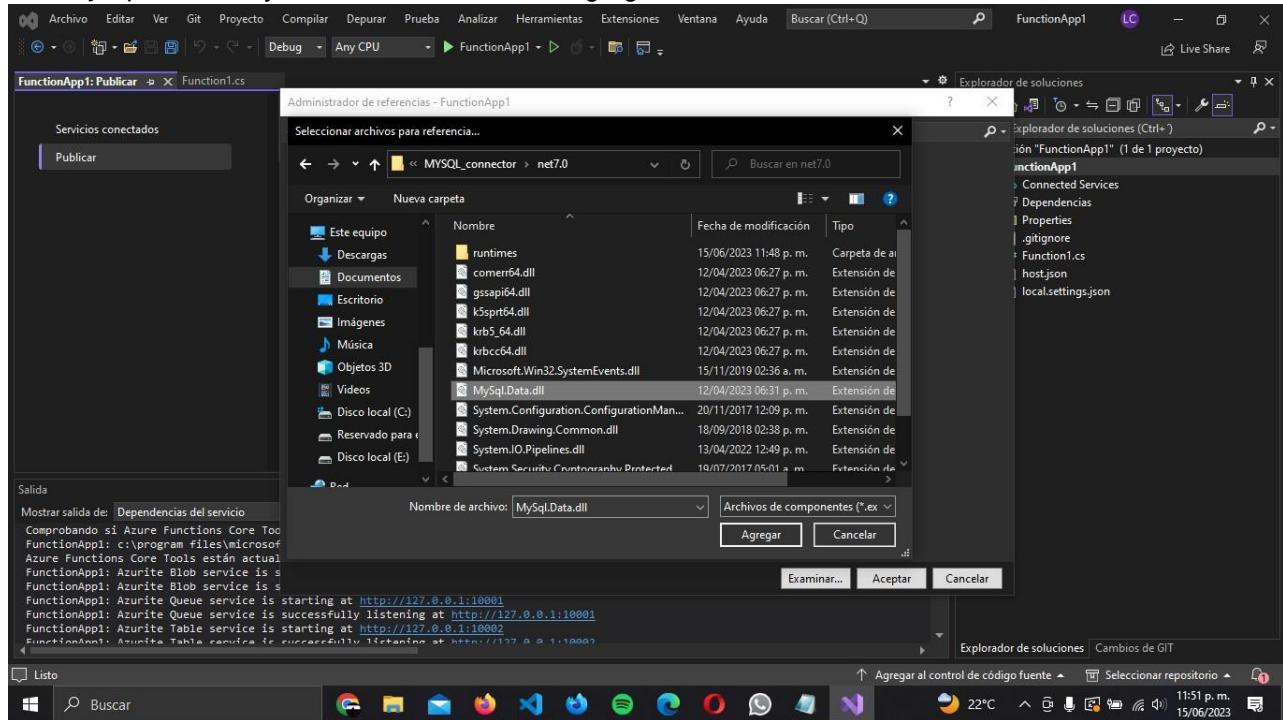


Figura 1.27. Instalación MYSQL Connector.

Y damos click en “Aceptar”.

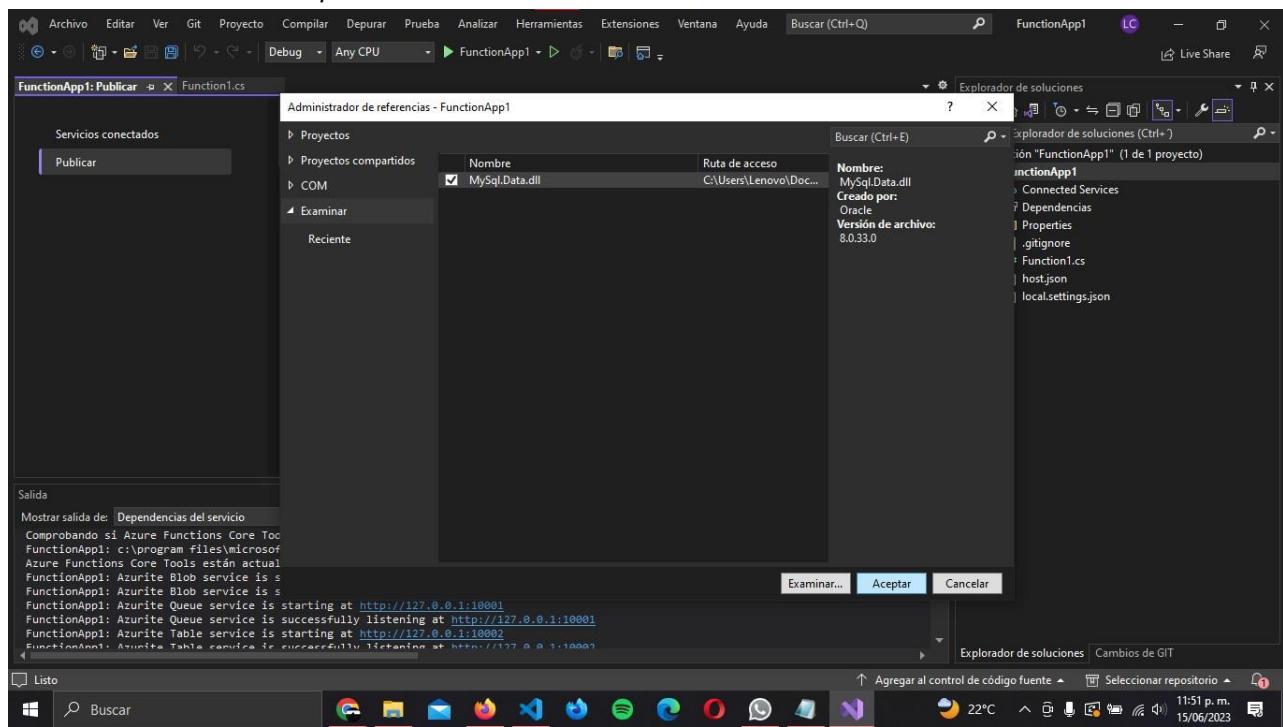


Figura 1.28. Instalación MYSQL Connector.

Dentro del portal de Azure, en nuestra aplicación de funciones, vamos al menú “Configuración” y damos click en la opción “+ Nueva configuración de la aplicación”.

The screenshot shows the Azure Functions configuration interface. The left sidebar has 'Configuración' selected. The main area shows application settings with their values. At the top right, there's a '+ Nueva configuración de la aplicación' button.

Nombre	Valor	Origen	Configuración de ranura de i...	Eliminar
APPLICATIONINSIGHTS_CONNECTION_STRING	<small>El valor está oculto. Haga clic para mostrar</small>	App Service	<input type="button" value=""/>	<input type="button" value=""/>
AzureWebJobsStorage	<small>El valor está oculto. Haga clic para mostrar</small>	App Service	<input type="button" value=""/>	<input type="button" value=""/>
FUNCTIONS_EXTENSION_VERSION	<small>El valor está oculto. Haga clic para mostrar</small>	App Service	<input type="button" value=""/>	<input type="button" value=""/>
FUNCTIONS_WORKER_RUNTIME	<small>El valor está oculto. Haga clic para mostrar</small>	App Service	<input type="button" value=""/>	<input type="button" value=""/>
WEBSITE_CONTENTAZUREFILECONNECTION	<small>El valor está oculto. Haga clic para mostrar</small>	App Service	<input type="button" value=""/>	<input type="button" value=""/>

Figura 1.29. Instalación MYSQL Connector.

Dentro del campo “Nombre” asignamos Server, y dentro del campo “Valor” escribimos “prueba-mysql.mysql.database.azure.com” y damos en aceptar.

The screenshot shows the 'Agregar o editar la configuración de la aplicación' dialog. It has fields for 'Nombre' (set to 'Server') and 'Valor' (set to 'prueba-mysql.mysql.database.azure.com'). There's also a checkbox for 'Configuración de ranura de implementación' which is unchecked. At the bottom are 'Aceptar' and 'Cancelar' buttons.

Figura 1.30. Instalación MYSQL Connector.

Damos click en la opción “Guardar” y damos click en el botón “Continuar”.

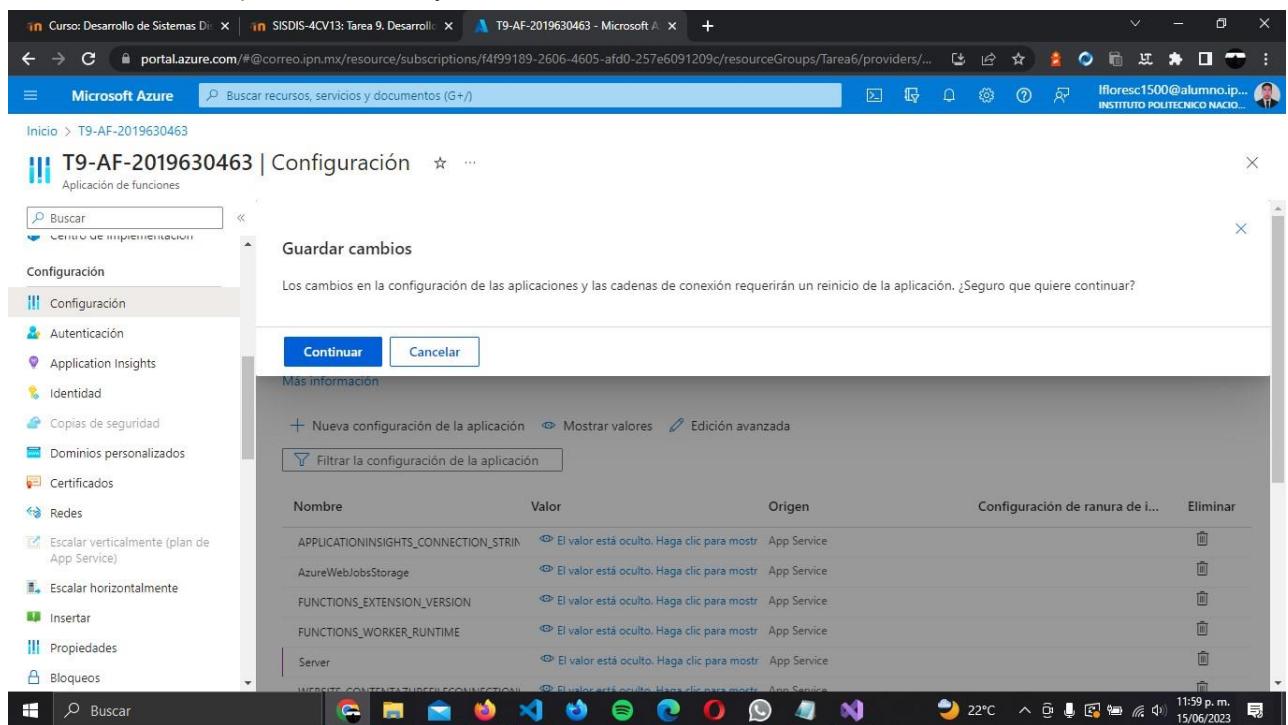


Figura 1.31. Instalación MYSQL Connector.

Podemos ver que los cambios se han guardado de forma correcta.

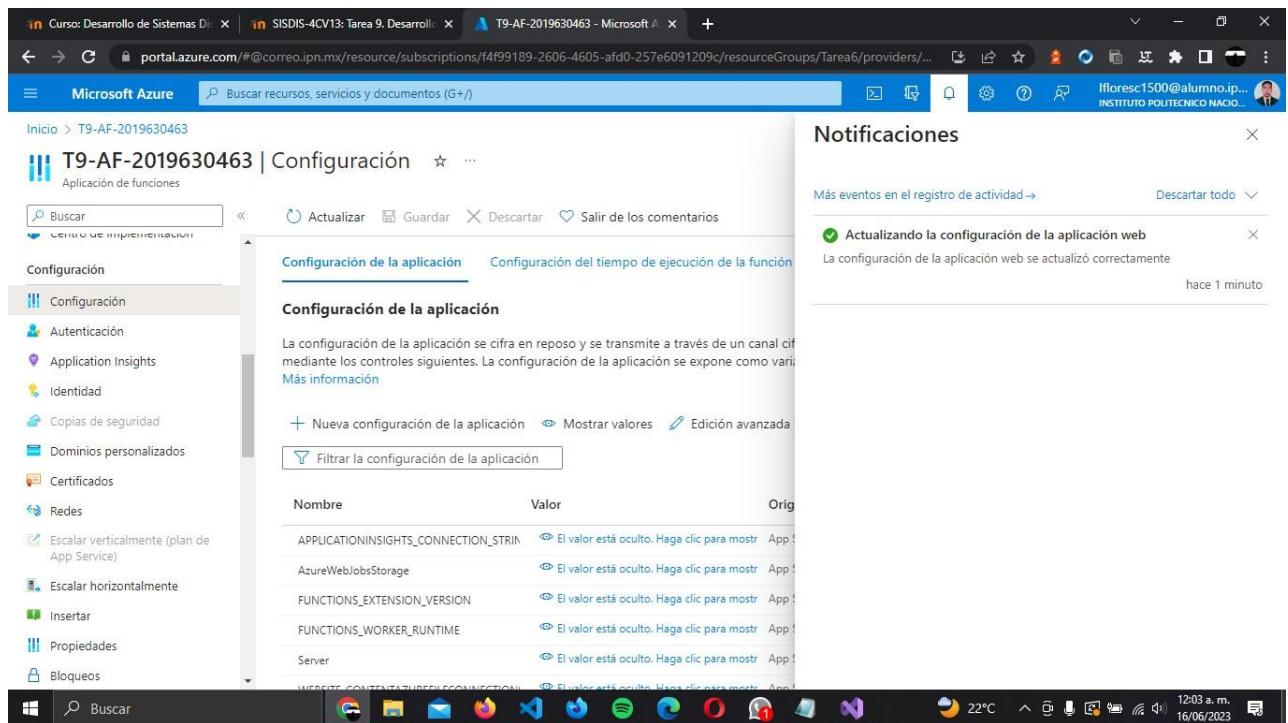


Figura 1.32. Instalación MYSQL Connector.

Ya instalado el conector, ahora allí mismo dentro de la aplicación de funciones, vamos al menú “CORS”.

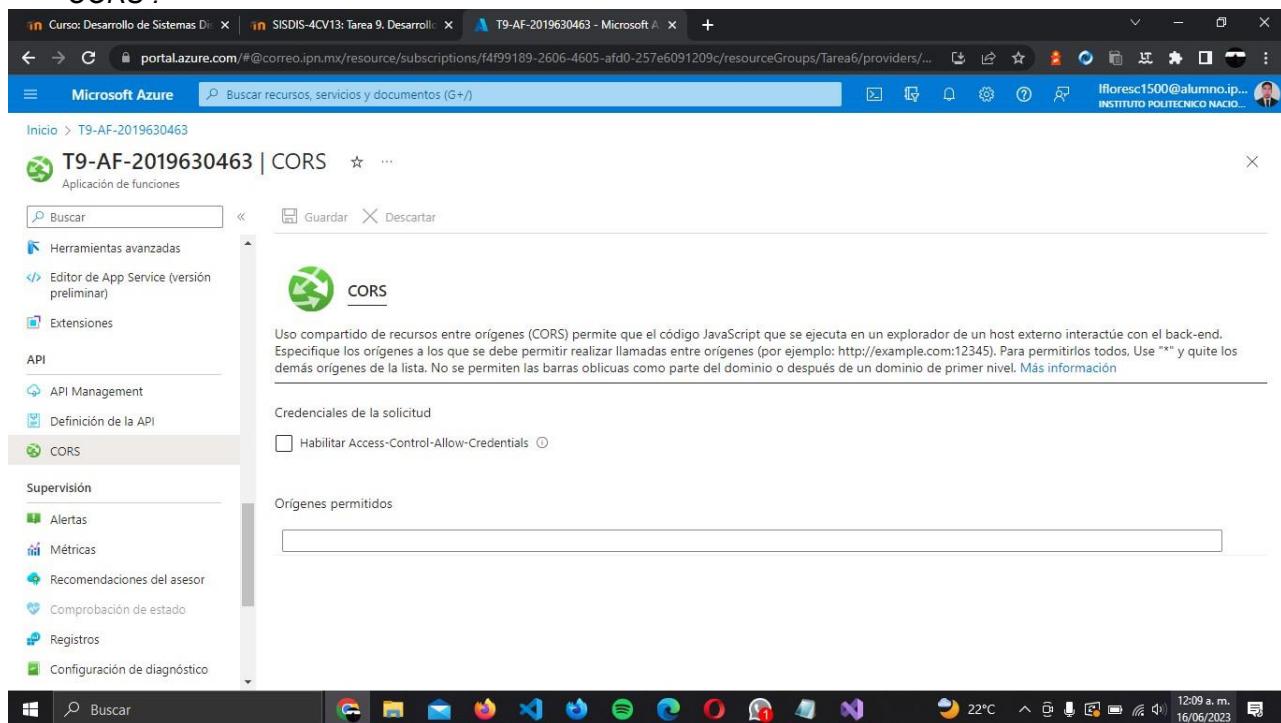


Figura 1.33. Configuración CORS.

Dentro del campo orígenes permitidos escribiremos “*” para habilitar todos los orígenes.

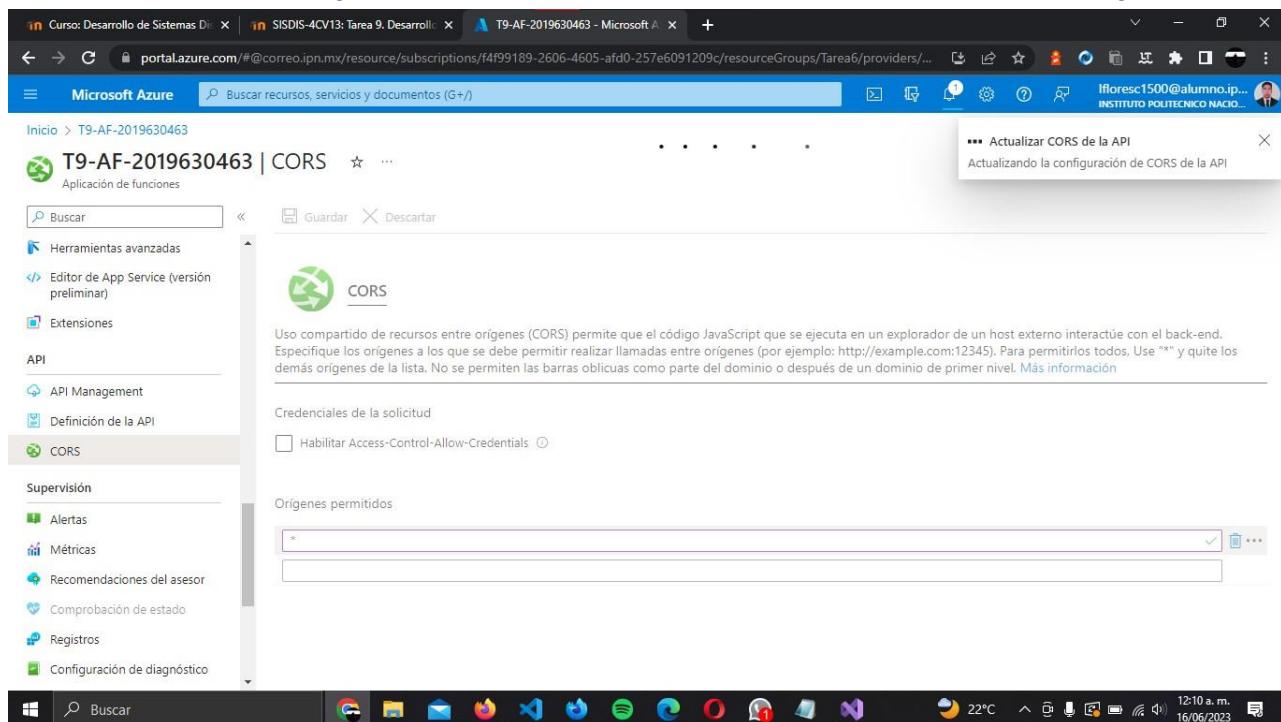


Figura 1.34. Configuración CORS.

Agregar una nueva función.

Ahora de nueva cuenta volvemos a Visual Studio 2022, crearemos una nueva función dentro de la aplicación de funciones. Daremos click derecho en nuestra aplicación de funciones que está en el “Explorador de soluciones”, después elegiremos el menú “Agregar” y la opción “Nueva función de Azure...”

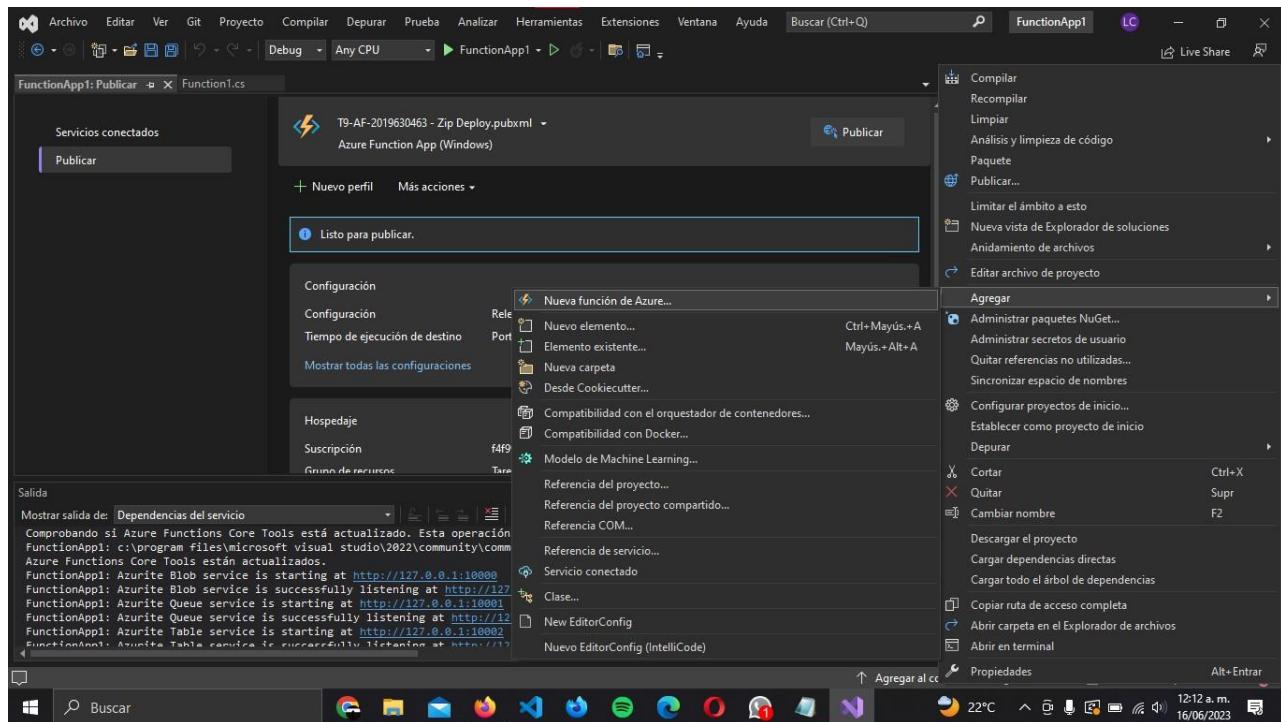


Figura 1.35. Creación de nueva función.

Se desplegará una pantalla y elegiremos la opción “Función de Azure” y damos click en “Agregar”.

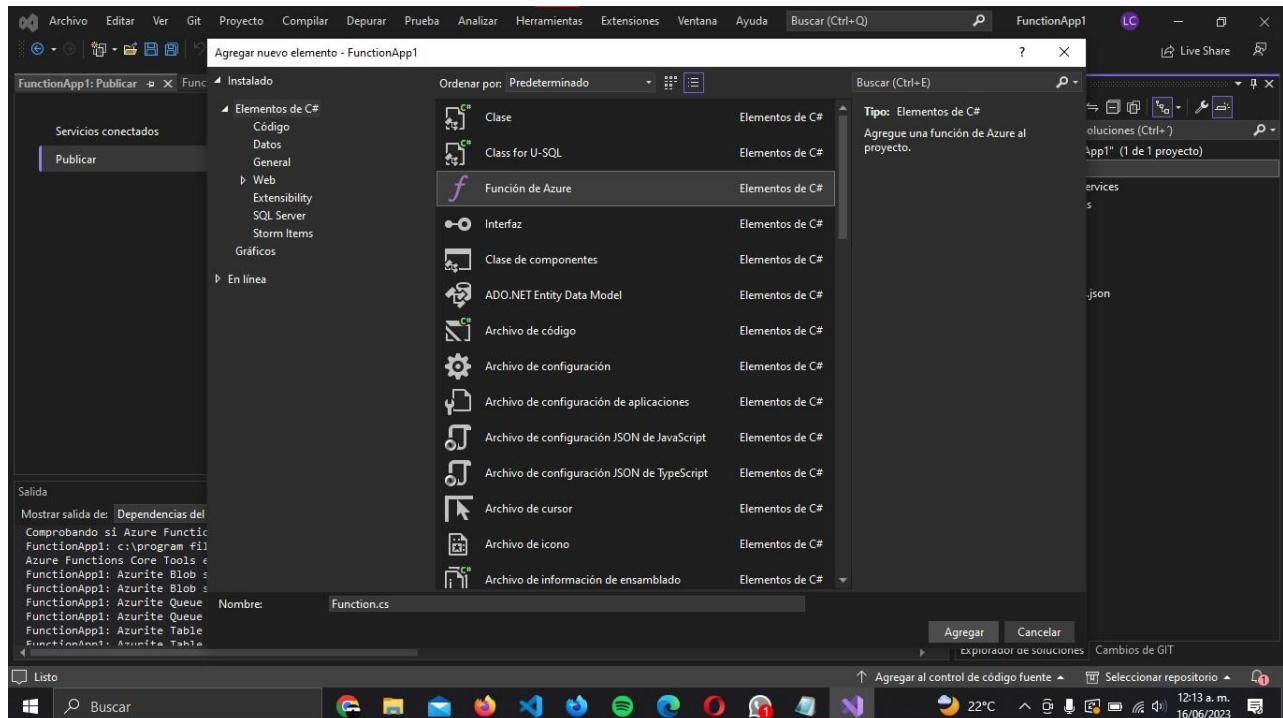


Figura 1.36. Creación de nueva función.

Se mostrará otra pantalla, dentro de esta elegiremos la opción “Http trigger” y damos click en “Agregar” y así se ha creado una nueva función

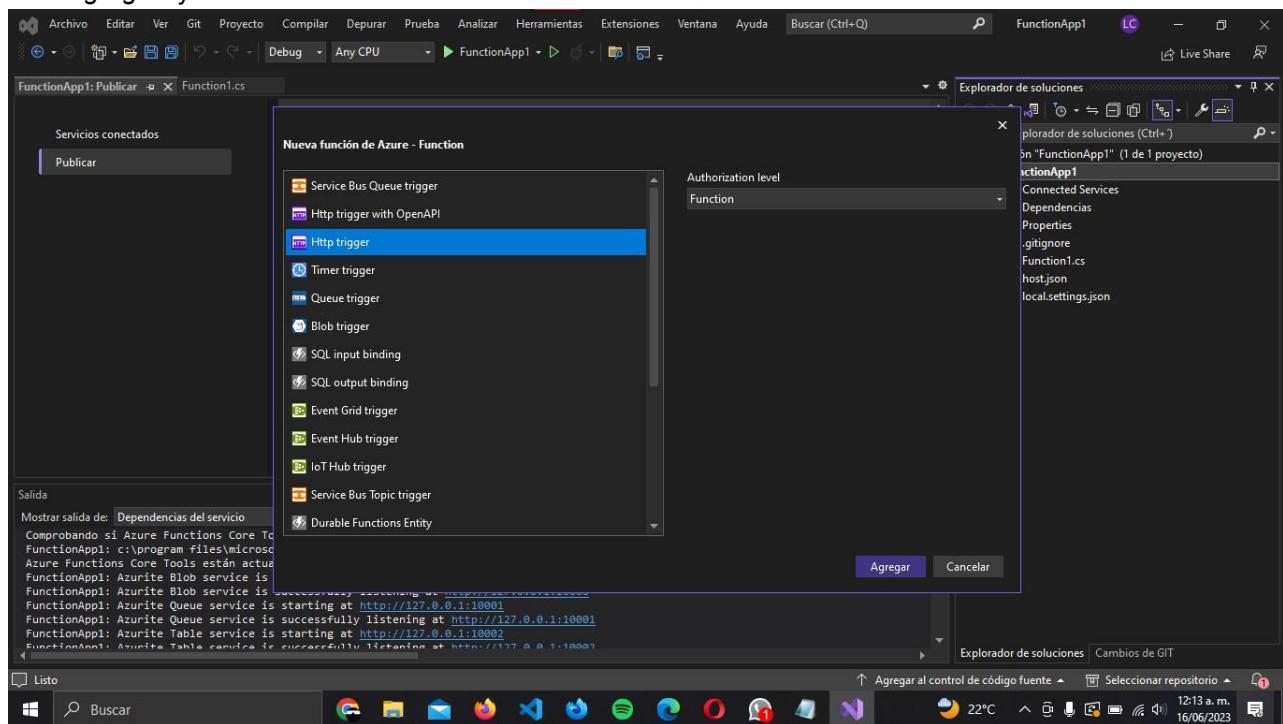


Figura 1.37. Creación de nueva función.

Creación de cuenta de almacenamiento.

Dentro del portal de Azure, seleccionamos la opción “Cuentas de almacenamiento”.

The screenshot shows the Microsoft Azure portal interface. In the top navigation bar, there are tabs for 'Curso: Desarrollo de Sistemas' and 'SISDIS-4CV13: Tarea 9. Desarroll...'. The main content area is titled 'Cuentas de almacenamiento' (Storage Accounts). On the left, there's a sidebar with 'Servicios de Azure' (Azure Services) and 'Recientes' (Recent) resources. The 'Cuentas de almacenamiento' service is selected. A modal window titled 'Cuentas de almacenamiento' is open, showing a brief description: 'Crea una cuenta de almacenamiento para conservar hasta 500 TB de datos en la nube. Use una cuenta de almacenamiento de uso general para guardar datos de objetos, utilice un almacén de datos de NoSQL, defina y us...' and three course links from Microsoft: 'Creación de una cuenta de Azure Storage', 'Conexión de una aplicación a Azure Storage...', and 'Protección de la cuenta de Azure Storage'. Below the modal, the 'Navegar' (Navigate) section shows a breadcrumb path: 'https://portal.azure.com/#blade/HubsExtension/BrowseResourceBlade/resourceType/Microsoft.Storage%2FStorageAccounts'. The taskbar at the bottom shows various pinned icons and the current date and time: '12:16 a. m. 16/06/2023'.

Figura 1.38. Creación de cuenta de almacenamiento.

Seleccionamos la cuenta de almacenamiento creada cuando se publicó el proyecto desde Visual Studio 2022.

The screenshot shows the 'Cuentas de almacenamiento' (Storage Accounts) list in the Microsoft Azure portal. The table displays one item: 'functionapp1202306152307'. The columns are 'Nombre' (Name), 'Tipo' (Type), 'Variante' (Variant), 'Grupo de recursos' (Resource Group), 'Ubicación' (Location), and 'Suscripción' (Subscription). The 'Nombre' column shows 'functionapp1202306152307', 'Tipo' shows 'Cuenta de almacenamiento', 'Variante' shows 'Storage', 'Grupo de recursos' shows 'Tarea6', 'Ubicación' shows 'Australia Central', and 'Suscripción' shows 'Azure for Students'. At the bottom, there are pagination controls: 'Página 1 de 1' and 'Siguiente >'. The taskbar at the bottom shows the URL 'https://portal.azure.com/#@correo.ipn.mx/resource/subscriptions/14f9189-2606-4605-af0-257e6091209c/resourceGroups/Tarea6/providers/Microsoft.Storage/storageAccounts/functionapp1202306152307' and the date and time '12:16 a. m. 16/06/2023'.

Figura 1.39. Creación de cuenta de almacenamiento.

Seleccionamos el menú “**Recursos compartidos de archivos**” y seleccionamos el file system de la aplicación de funciones.

Nombre	Modificado	Cuota
t9-af-2019630463	15/6/2023, 23:10:48	5 TiB

Figura 1.40. Creación de cuenta de almacenamiento.

Dentro del file system, ahora seleccionamos el menú “**Examinar**” y damos click en la opción “**Cargar**”.

Nombre	Tipo	Tamaño
ASP.NET	Directorio	***
data	Directorio	***
LogFiles	Directorio	***
site	Directorio	***

Figura 1.41. Creación de cuenta de almacenamiento.

Cargamos dos archivos, en este caso el main.html y WSClient.js.

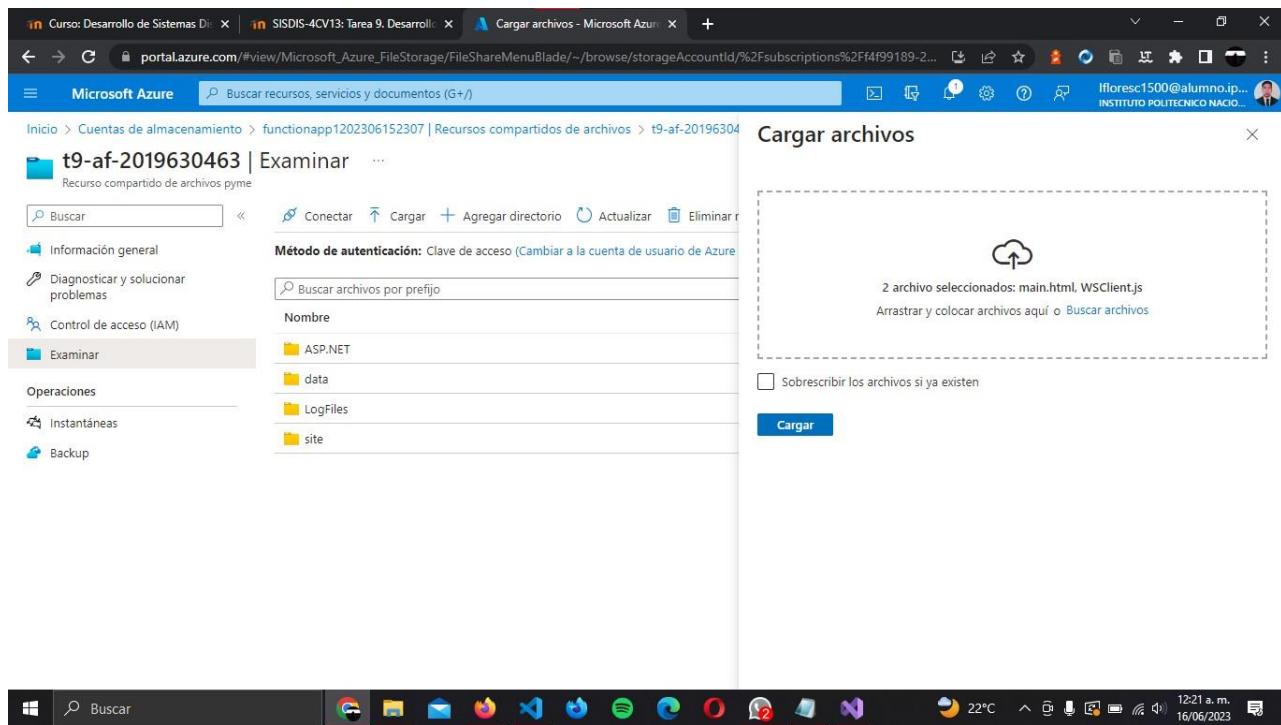


Figura 1.42. Creación de cuenta de almacenamiento.

Podemos ver que los archivos fueron cargados de manera correcta.

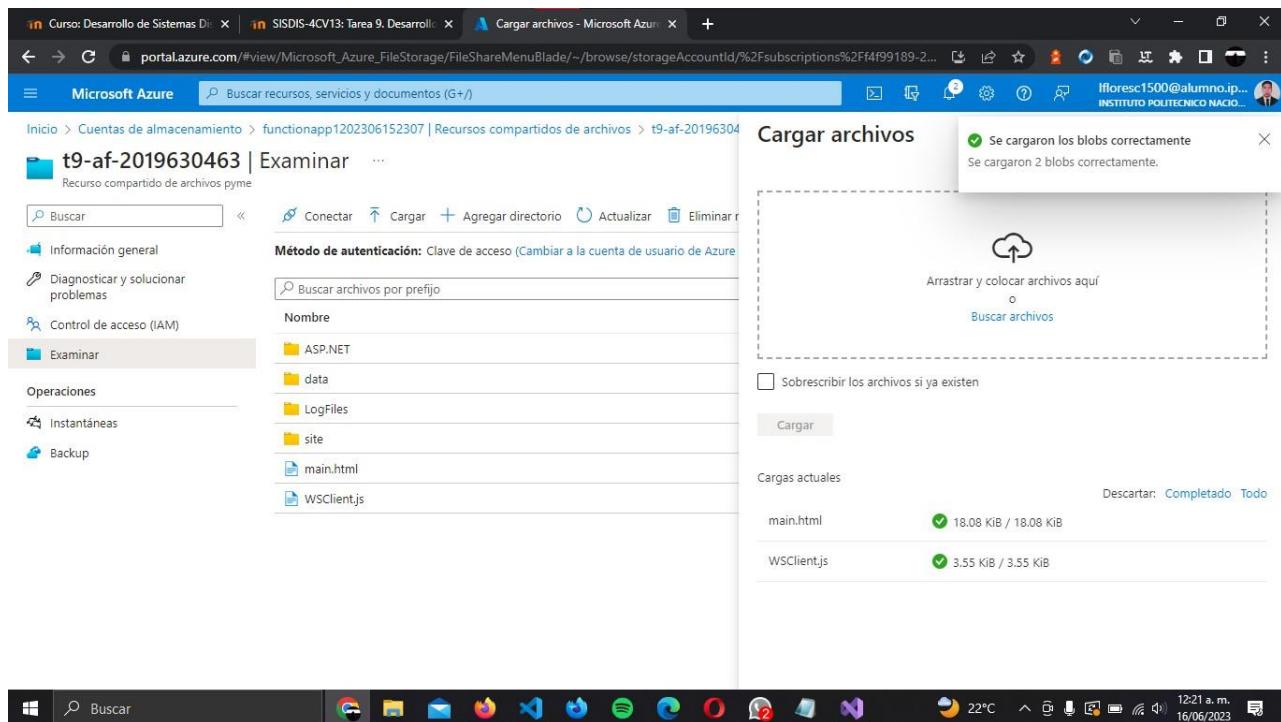


Figura 1.43. Creación de cuenta de almacenamiento.

Creación de instancia MySQL.

Dentro del portal de Azure, ingresamos “*Servidores de Azure Database for MySQL*” en la barra de búsqueda y damos click en la opción que se muestra.

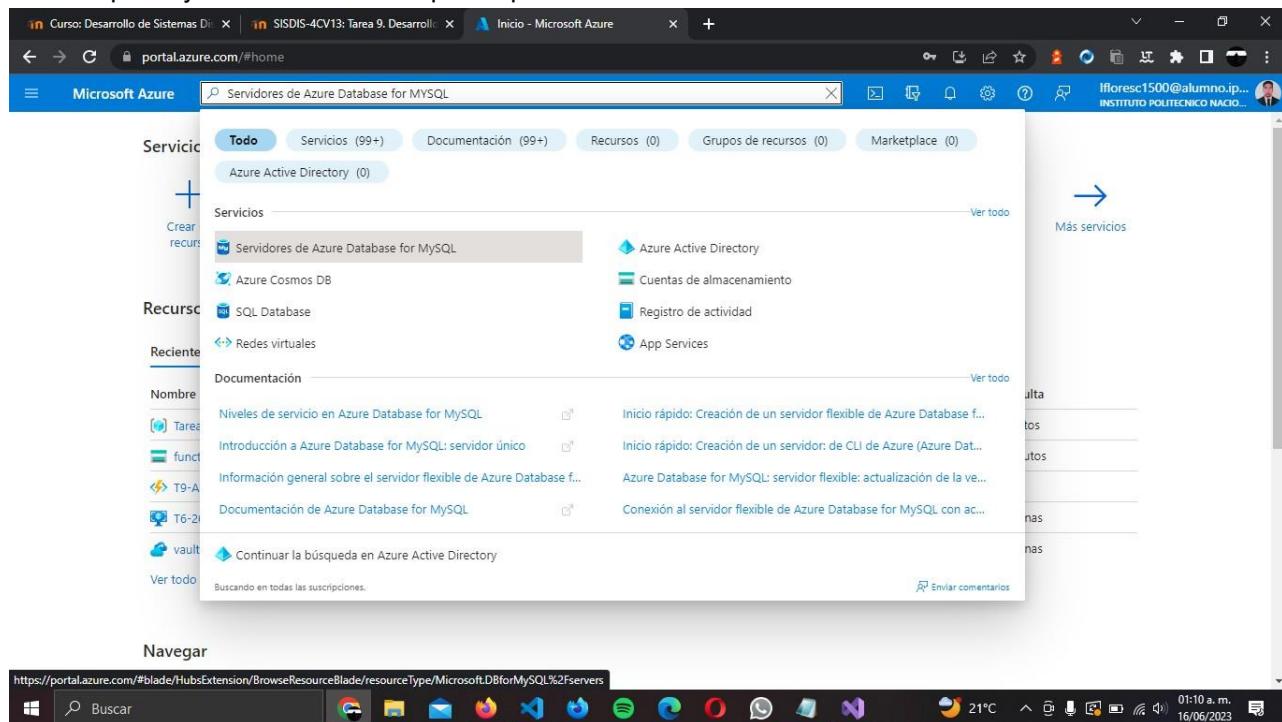


Figura 1.44. Creación de instancia MySQL.

Damos click en el botón “Crear Database para el servidor MySQL”

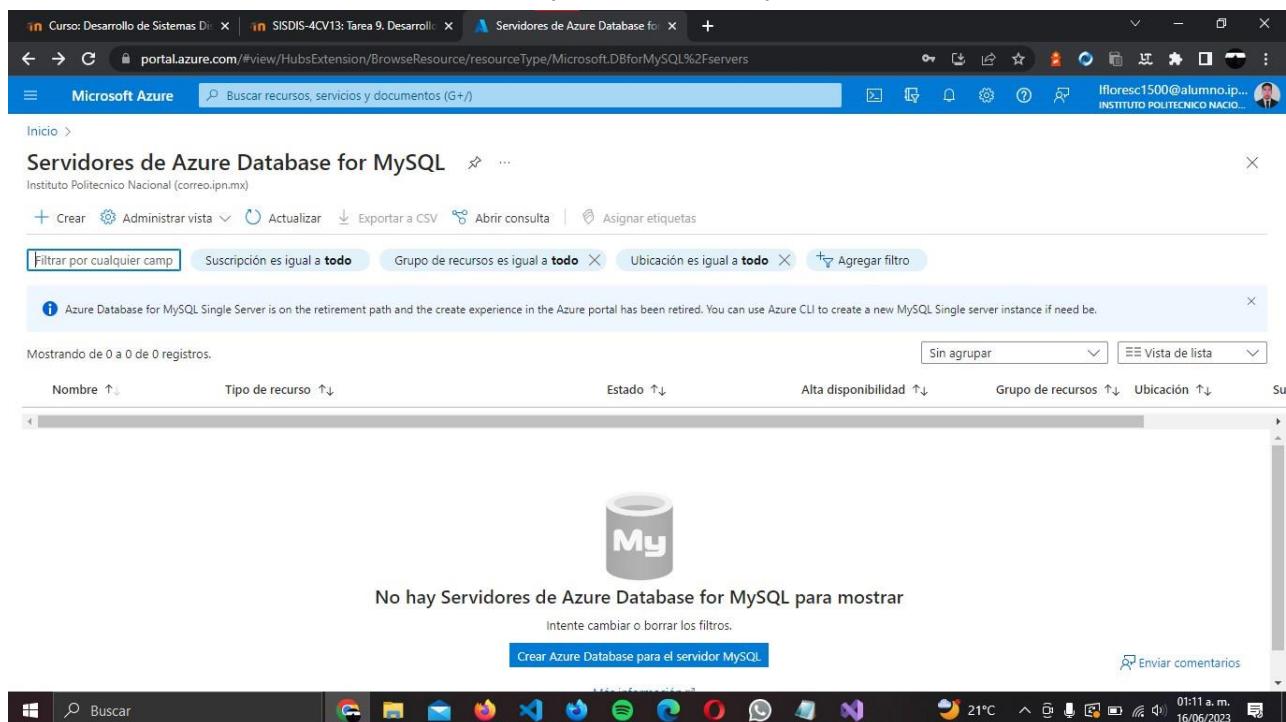


Figura 1.45. Creación de instancia MySQL.

Se mostrarán dos opciones damos click en “Crear” dentro de “Servidor flexible”.

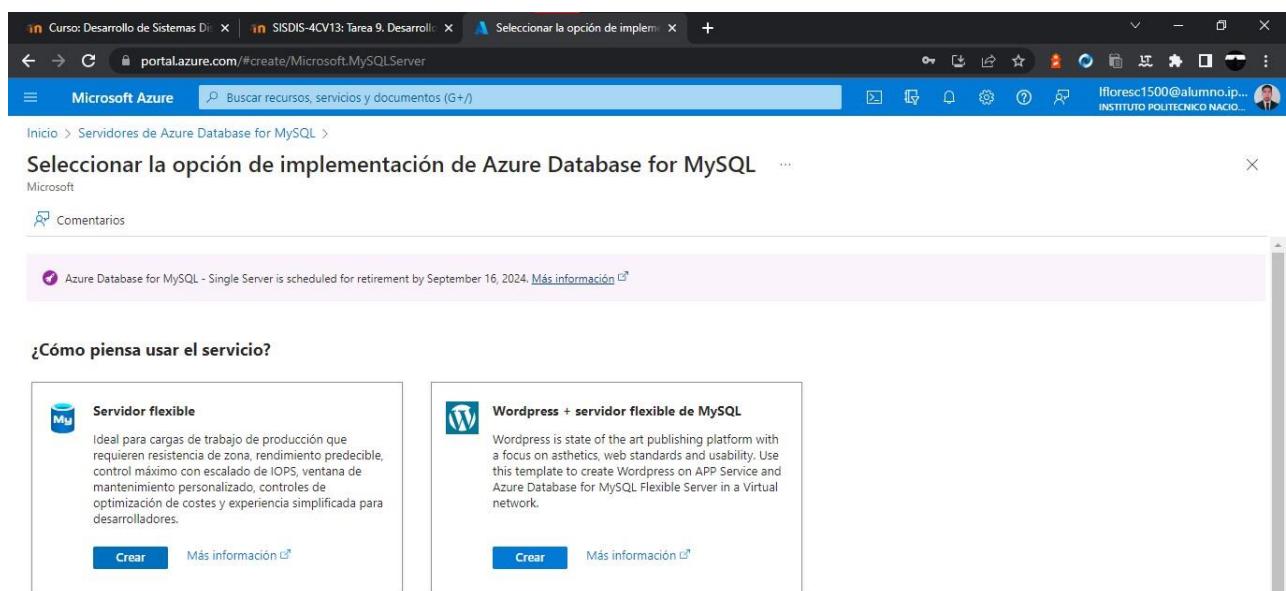


Figura 1.46. Creación de instancia MySQL.

Dentro de la configuración, primero elegiremos el grupo de recursos, en este caso “Tarea6”.

Servidor flexible

Básico Redes Seguridad Etiquetas Revisar y crear

Cree un servidor flexible de Azure Database for MySQL. [Más información](#)

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * [Azure for Students](#)

Grupo de recursos * [Tarea6](#)

[Revisar y crear](#) [Siguiente: Redes >](#)

Costes estimados

- SKU de proceso** USD 6.21/mes
 - Liberar hasta 750 horas Standard_B1s (1 vCore) 6.21
- Almacenamiento** USD 2.30/mes
 - Liberar hasta 32 GB Almacenamiento seleccionado 20 x 20 GB (USD0.12 por GiB) 0.12
- Retención de la copia de seguridad**
 - La retención de copias de seguridad

Figura 1.47. Creación de instancia MySQL.

Ahora ingresamos el nombre del servidor, en este caso “T9-BD-2019630463”.

Servidor flexible

Nombre del servidor * T9-BD-2019630463

Región * East US

Versión de MySQL * 5.7

Tipo de carga de trabajo Para proyectos de desarrollo o aficiones

Proceso y almacenamiento Con capacidad de ráfaga, B1s

[Revisar y crear](#) [Siguiente: Redes >](#)

Costes estimados

- SKU de proceso** USD 6.21/mes
 - Liberar hasta 750 horas Standard_B1s (1 vCore) 6.21
- Almacenamiento** USD 2.30/mes
 - Liberar hasta 32 GB Almacenamiento seleccionado 20 x 20 GB (USD0.12 por GiB) 0.12
- Retención de la copia de seguridad**
 - La retención de copias de seguridad

Figura 1.48. Creación de instancia MySQL.

Como se pudo observar en la figura anterior al ingresar el nombre “T9-BD-2019630463”, se mostraban distintos errores, como que el nombre debe estar completamente en minúsculas y debe estar disponible, por lo que se tuvo que asignar el nombre “t9-bd-2019630463”.

En el campo “Region” se eligió “East US”.

En el campo “Versión de MySQL” se eligió la versión 8.0

En el campo “Tipo de carga de trabajo” se seleccionó “Para proyectos de desarrollo o aficiones”.

En el campo “Proceso y almacenamiento” damos click en la opción “Configurar servidor”.

The screenshot shows the Microsoft Azure portal interface for creating a MySQL server. The main form is titled "Servidor flexible" and contains the following fields:

- Nombre del servidor: t9-bd-2019630463
- Región: East US
- Versión de MySQL: 8.0
- Tipo de carga de trabajo: Para proyectos de desarrollo o aficiones
- Proceso y almacenamiento: Con capacidad de ráfaga, B1s (1 núcleo virtual, 1 GB de RAM, 20 GB de almacenamiento, 360 IOPS). Redundancia geográfica: Disabled.

A sidebar titled "Costes estimados" provides a breakdown of the estimated costs:

Categoría	Detalle	Coste
SKU de proceso	Liberar hasta 750 horas Standard_B1s (1 vCore)	6.21
	Almacenamiento seleccionado 20 x 20 GB (USD0.12 por GB)	0.12
Almacenamiento	USD 2.30/mes	
Retención de la copia de seguridad	La retención de copias de seguridad	

At the bottom, there are buttons for "Revisar y crear" and "Siguiente: Redes >". The taskbar at the bottom of the screen shows various application icons and the system clock.

Figura 1.49. Creación de instancia MySQL.

Dentro de esta opción realizamos las siguientes configuraciones.

En el campo “*Computer tier*” se seleccionó “*Flexible*”.

En el campo “*Tipo de proceso*” se seleccionó “*Standard_B1s*”.

En el campo “*Storage size (in GiB)*” se seleccionó “20”.

En el campo “*IOPS*” se seleccionó “*IOPS aprovisionados previamente*”.

Finalmente se activó la casilla “*Crecimiento automático del almacenamiento*”.

Damos click en el botón “*Guardar*”.

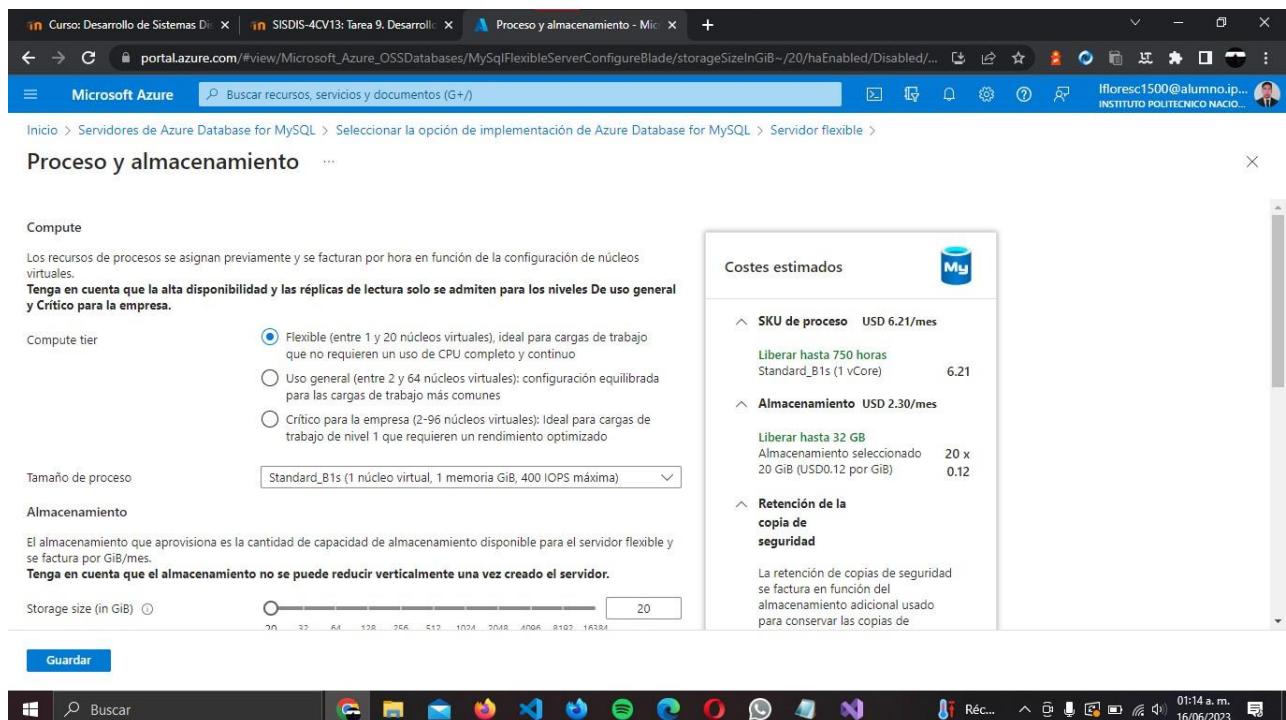


Figura 1.50. Creación de instancia MySQL.

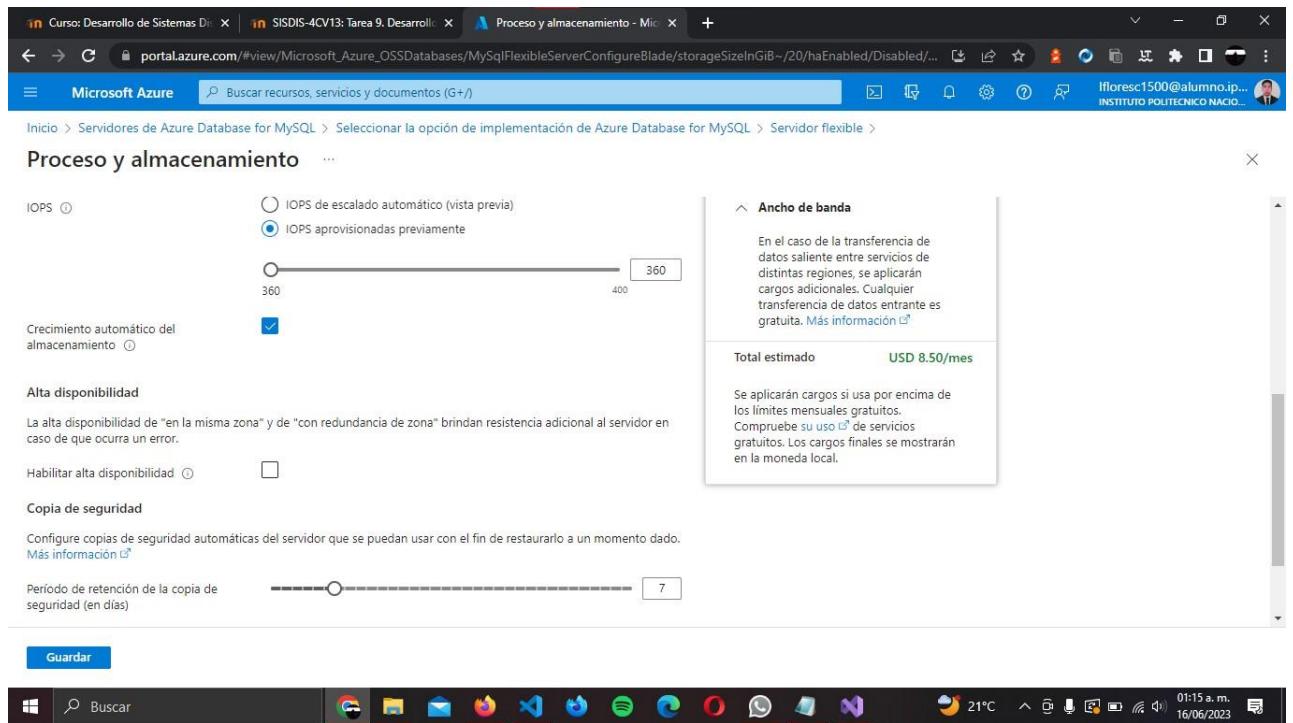


Figura 1.51. Creación de instancia MySQL.

Después de haber guardado la configuración del servidor, ahora elegimos método de autenticación la “Autenticación de MySQL” e ingresamos tanto nombre como contraseña.

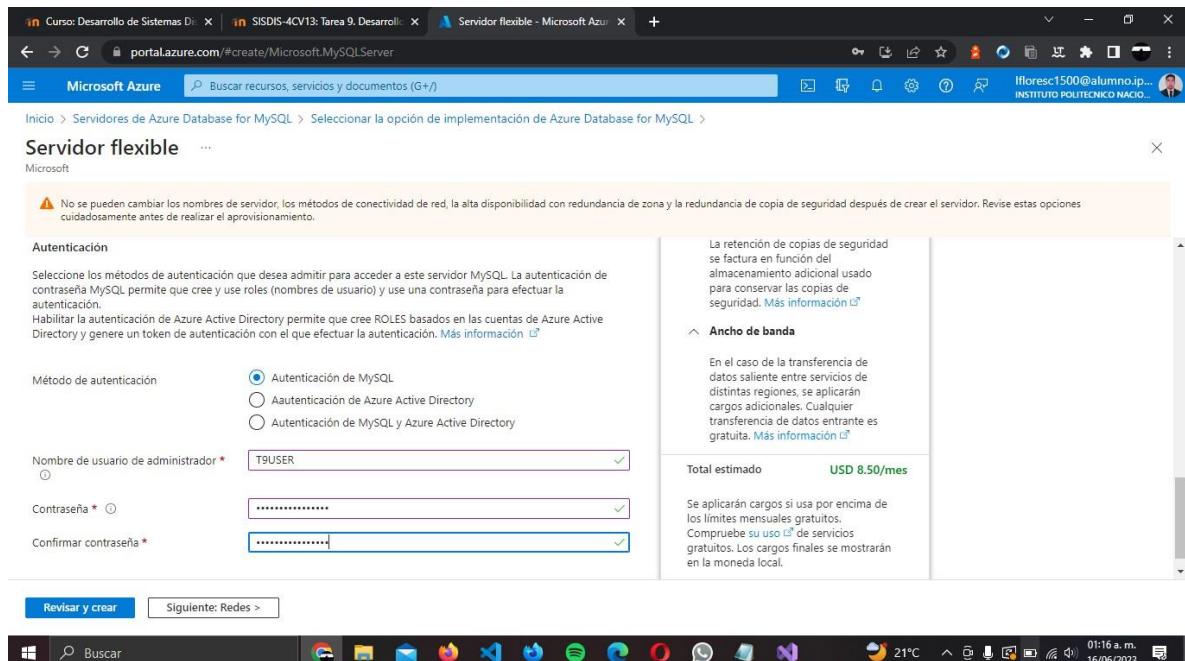


Figura 1.52. Creación de instancia MySQL.

Pasando al menú de “**Redes**” seleccionamos “**Acceso público (direcciones IP permitidas)**” como Método de conectividad.

Servidor flexible

Redes

Configure el acceso de redes y la seguridad para su servidor.

Conectividad de red

Puede conectarse a su servidor especificando una dirección IP pública a continuación, o bien mediante una red virtual seleccionada.

Método de conectividad

- Acceso público (direcciones IP permitidas)
- Acceso privado (integración con red virtual)

Las conexiones procedentes de las direcciones IP configuradas en la sección Reglas de firewall, más abajo, tendrán acceso a este servidor. De forma predeterminada, no se permiten direcciones IP públicas. [Más información](#)

Costes estimados

- SKU de proceso** USD 6.21/mes
- Almacenamiento** USD 2.30/mes
- Retención de la copia de seguridad**

Revisar y crear < Anterior Siguiente: Seguridad >

Figura 1.53. Creación de instancia MySQL.

Dentro de este mismo menú activamos la casilla “**Permitir acceso público a este servidor desde cualquier servicio de Azure dentro de Azure**” y agregamos la IP haciendo click en “**+ Agregar**”.

Servidor flexible

Reglas de firewall

Se permitirán las conexiones entrantes desde las direcciones IP especificadas a continuación en el puerto 3306 de este servidor. [Más información](#)

Permitir acceso público a este servidor desde cualquier servicio de Azure dentro de Azure

+ Agregar dirección IP del cliente actual (189.203.98.171) + Agregar 0.0.0.0 - 255.255.255.255

Nombre de la regla de firewall	Dirección IP inicial	Dirección IP final
ClientIPAddress_2023-6-16_1-17-53	189.203.98.171	189.203.98.171
Nombre de la regla de firewall	Dirección IP inicial	Dirección IP final

Conexiones cifradas

Este servidor admite conexiones cifradas que usan la seguridad de la capa de transporte (TLS 1.2). Para obtener más detalles sobre la descarga del certificado, consulte la información sobre “conexión mediante TLS/SSL”. [Más información](#)

Ancho de banda

En el caso de la transferencia de datos saliente entre servicios de distintas regiones, se aplicarán cargos adicionales. Cualquier transferencia de datos entrante es gratuita. [Más información](#)

Total estimado USD 8.50/mes

Se aplicarán cargos si usa por encima de los límites mensuales gratuitos. Compruebe [su uso](#) de servicios gratuitos. Los cargos finales se mostrarán

Revisar y crear < Anterior Siguiente: Seguridad >

Figura 1.54. Creación de instancia MySQL.

Damos click en el botón “*Revisar y crear*” y vemos los detalles del servidor, damos click en el botón “*Crear*”.

Figura 1.55. Creación de instancia MySQL.

Podemos observar que la implementación del servidor está en curso.

Figura 1.56. Creación de instancia MySQL.

Se puede observar que la instancia ha sido creada, así que damos click en el botón “Ir al recurso”.

The screenshot displays the Microsoft Azure portal interface. In the center, a card for a MySQL instance named "MySQLFlexibleServer_b40b29c5b5c9432aaabd96c8f558ab1e" is shown. The "Información general" tab is active, indicating that the implementation was successful ("Se completó la implementación"). To the right, a "Notificaciones" panel shows a successful deployment message. At the bottom of the main card, there are links for "Enviar comentarios" and "Cuéntenos su experiencia con la implementación". The taskbar at the bottom of the screen shows various application icons, including a search bar, file explorer, email, browser, and other system utilities. The date and time in the bottom right corner indicate it's June 16, 2023, at 10:27 a.m.

Figura 1.57. Creación de instancia MySQL.

Finalmente podemos ver los detalles de la instancia ya creada.

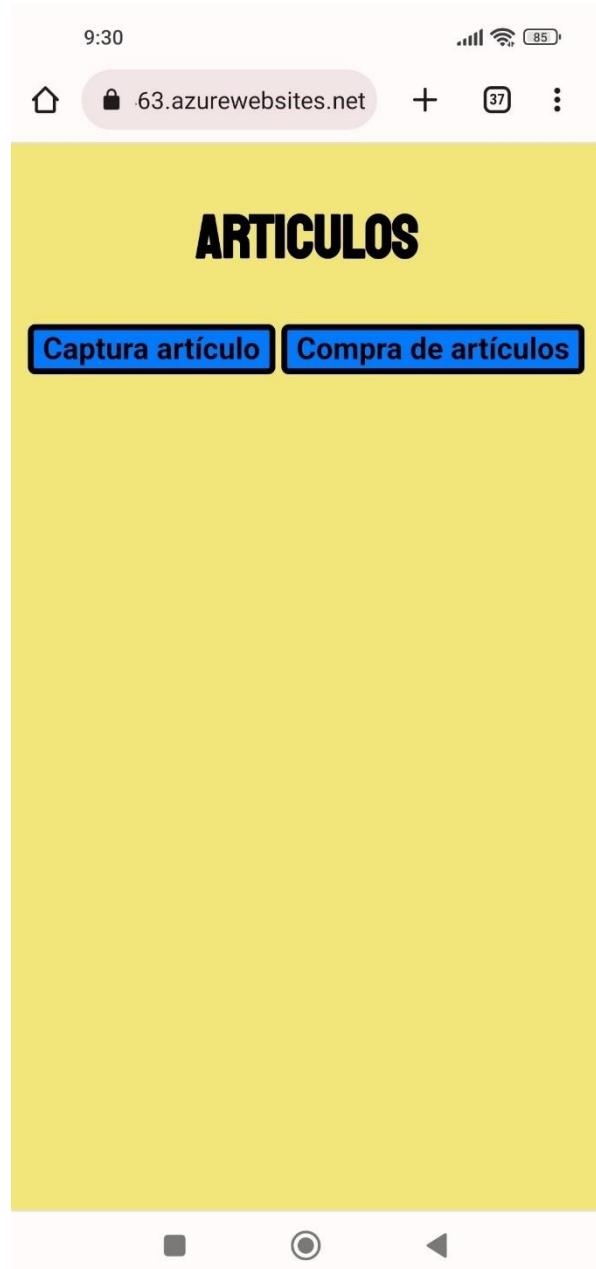
The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes tabs for 'Curso: Desarrollo de Sistemas Di...', 'SISDIS-4CV13: Tarea 9. Desarroll...', and 't9-bd-2019630463 - Microsoft A...'. The main title is 't9-bd-2019630463' with the subtitle 'Servidor flexible de Azure Database for MySQL'. The left sidebar contains navigation links such as 'Información general', 'Registro de actividad', 'Control de acceso (IAM)', 'Etiquetas', 'Diagnosticar y solucionar problemas', 'Learning center', 'Configuración', 'Proceso y almacenamiento', 'Redes', 'Bases de datos', 'Conectar', 'Parámetros del servidor', 'Replicación', and 'Mantenimiento'. The main content area displays the 'Información general' section for the MySQL instance, listing details like subscription (Azure for Students), resource group (Tarea6), location (East US), and creation date (2023-06-16). It also includes sections for 'Introducción', 'Propiedades', 'Recomendaciones', 'Supervisión', and 'Tutoriales'. At the bottom, there are buttons for 'Learn', 'Permitir acceso', and 'Conectar', along with a search bar and system status indicators.

Figura 1.58. Creación de instancia MySQL.

Pruebas de funcionamiento en dispositivo móvil.

R1: El cliente deberá desplegar inicialmente un menú con dos opciones: Captura de artículo y Compra de artículos.

Podemos ver que se despliega la pantalla principal, mostrando las dos opciones “Captura artículo” y “Compra de artículos”, primero realizaremos la captura de 2 artículos, así que damos clic sobre el botón “*Captura artículo*”.



R2: Al seleccionar la opción "Captura de artículo" el cliente deberá desplegar la pantalla "Captura de artículos" la cual deberá permitir capturar el nombre del artículo, la descripción del artículo, el precio, la cantidad en almacén y la fotografía del artículo. Los datos de los artículos se deberán guardar en una tabla llamada "articulos". Cada artículo deberá tener un ID auto-incremental.

Después de haber seleccionado "Captura de artículo", se despliega una pantalla en la cual se deben ingresar los datos del artículo.

9:31

63.azurewebsites.net

CAPTURA DE ARTÍCULOS

Nombre artículo
Cheetos torciditos

Descripción artículo
Sabritas Cheetos torciditos sabor queso y Chile 240 g

Precio artículo
46.50

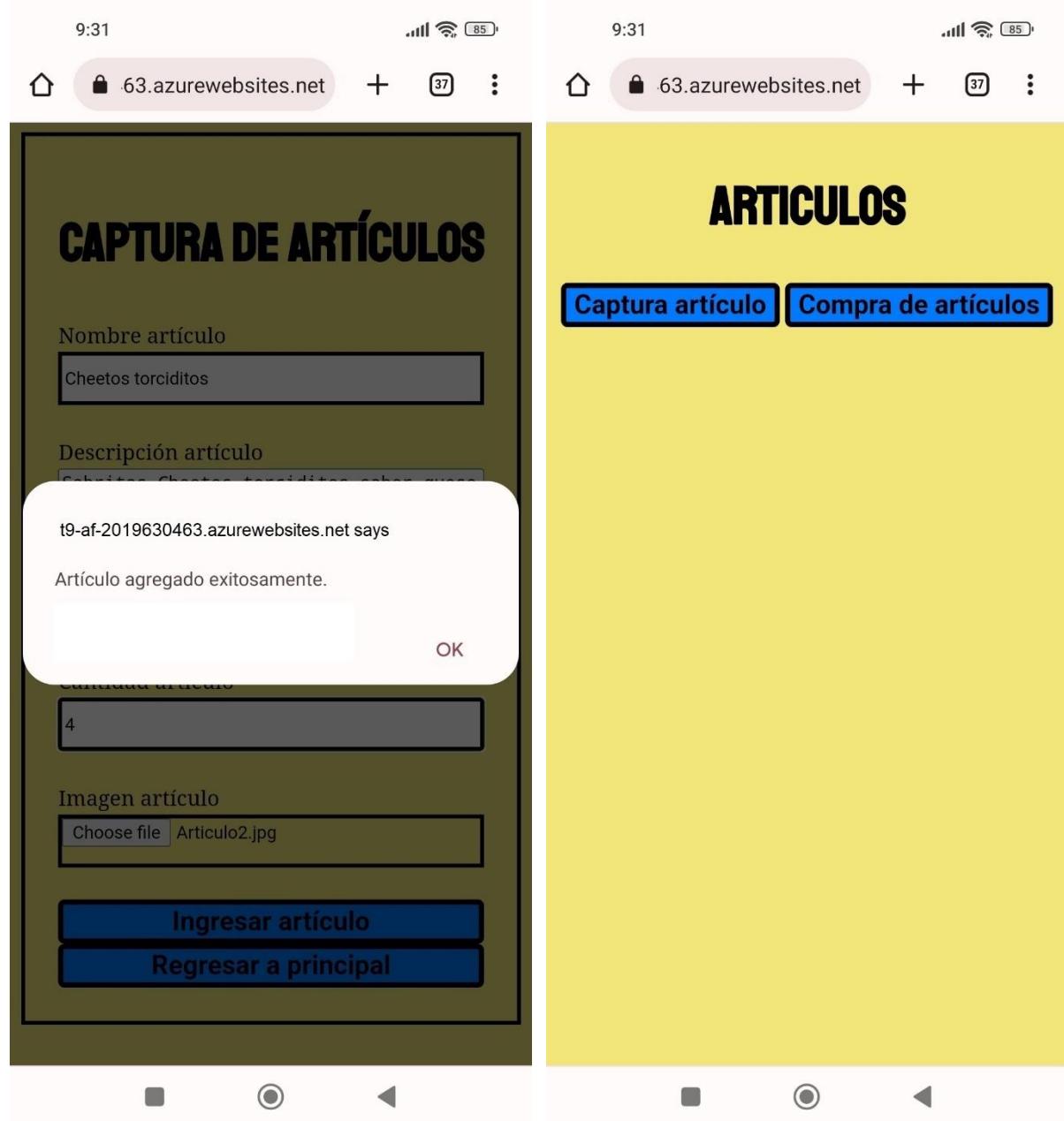
Cantidad artículo
4

Imagen artículo
Choose file Articulo2.jpg

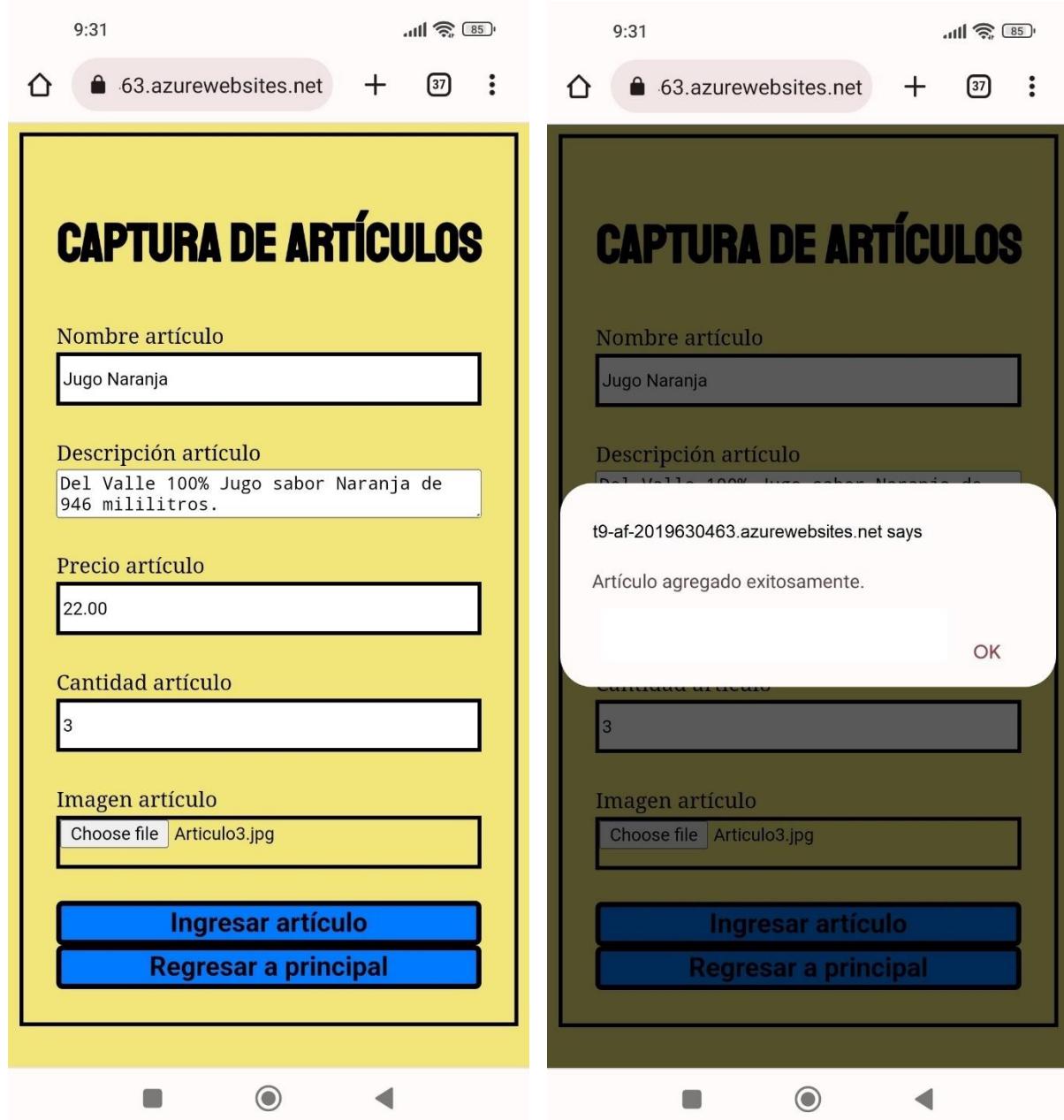
Ingresar artículo

Regresar a principal

Para ingresarlo ahora seleccionamos el botón “Ingresar artículo” y se despliega una ventana que indica que ha sido agregado correctamente, después se redirecciona de nuevo a la página principal.

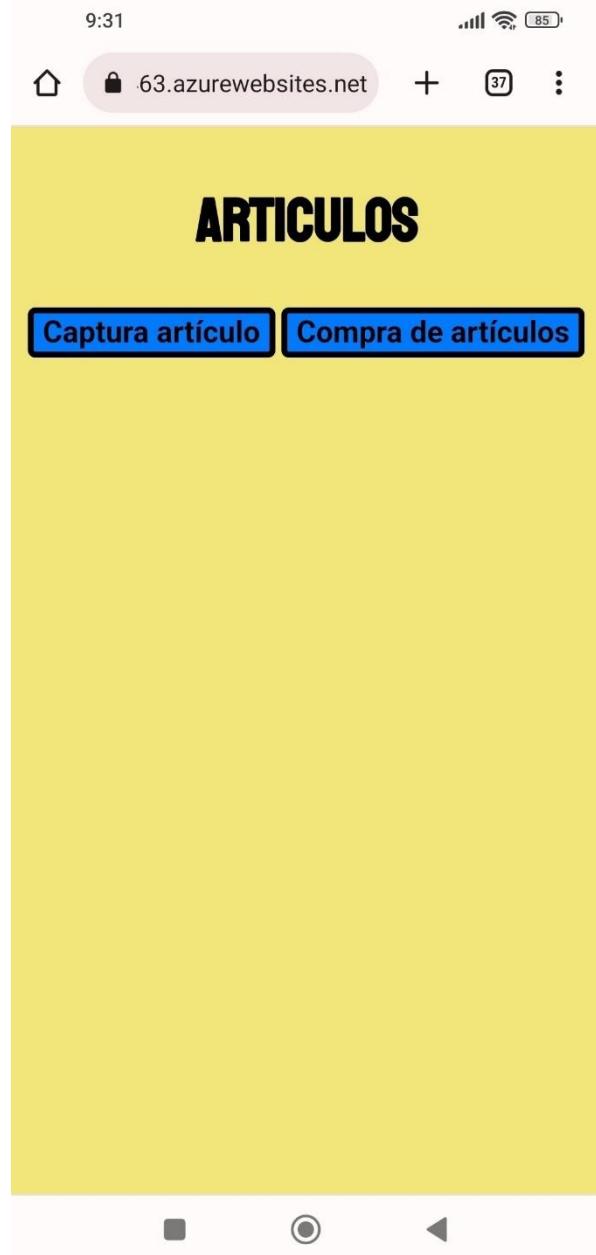


Después agregamos el segundo artículo siguiendo los mismos pasos y se agrega de forma exitosa.



R3: 3. Al seleccionar la opción "Compra de artículos" el cliente deberá desplegar la pantalla "Compra de artículos" la cual deberá permitir al usuario buscar artículos ingresando una palabra la cual se buscará en los campos "nombre" y "descripcion" de la tabla "articulos". La búsqueda se deberá realizar utilizando una instrucción SELECT con LIKE.

Ahora se comprará un artículo, primero seleccionamos el botón “Compra de artículos”,



Se desplegará una pantalla “Compra de artículos”, ahora se realizará la búsqueda de un producto mediante su nombre y descripción.

R4: Los datos de los artículos (fotografía, nombre y precio) que resulten de una búsqueda se deberán desplegar en la pantalla “Compra de artículos”.

Y podemos ver que el artículo buscado muestra nombre, precio, cantidad y su fotografía.



R5: Para cada artículo resultado de la búsqueda, se deberá desplegar un botón de "Compra" y un campo de "Cantidad" con un valor default igual a 1. Para cada artículo se deberá desplegar un botón "Descripción", al dar clic a este botón se deberá desplegar una ventana con la descripción del artículo y un botón "Cerrar" para cerrar esta ventana.

Como podemos ver al buscar el artículo, se despliegan la cantidad del artículo como 1 por default, un botón "Comprar" y "Ver descripción", si hacemos clic en "Ver descripción" se desplegará una ventana con la descripción del producto, y la cerramos con el botón "Cerrar."

The image consists of two side-by-side screenshots of a mobile application interface.

Screenshot 1 (Left): The title is "COMPRA DE ARTICULOS". It shows a search bar with "Cheetos torciditos" and "torciditos" entered. A blue button labeled "Buscar artículo" is visible. Below the search bar are two buttons: "Carrito de compras" (blue background) and "Regresar a principal" (white background). The main section is titled "ARTICULOS ENCONTRADOS" and displays a single product item. The product image is a bag of "Cheetos Torciditos" cheese and chile flavor. Below the image, the product details are listed: "Nombre: Cheetos torciditos", "Precio: 46.50", and "Cantidad: 1". At the bottom are two blue buttons: "Comprar" and "Ver descripción".

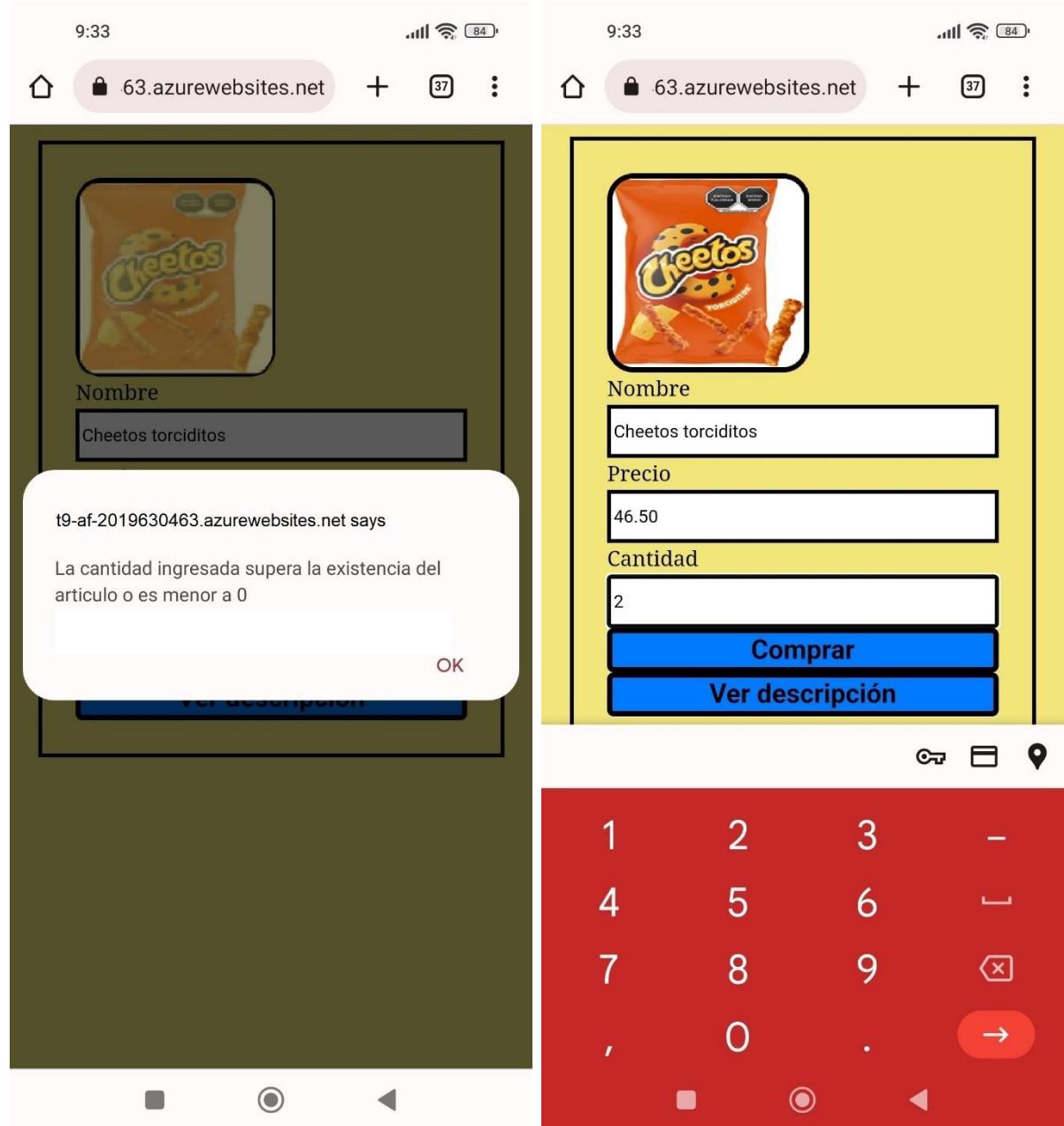
Screenshot 2 (Right): The title is "Descripción del artículo". The content is "Sabritas Cheetos torciditos sabor queso y chile 240 g". At the bottom is a small "Cerrar" button.

R6: Cuando el usuario presione el botón de "Compra", si la cantidad de artículos a comprar es menor o igual a la cantidad de artículos en la tabla "articulos", se deberá insertar en una tabla llamada "carrito_compra" el ID del artículo y la cantidad, así mismo se restará la cantidad solicitada de la cantidad en la tabla de "artículos". Si la cantidad de artículos a comprar es mayor a la cantidad de artículos en la tabla "articulos" se deberá desplegar un mensaje indicando al usuario el número de artículos disponibles. El INSERT a la tabla "carrito_compra" y el UPDATE a la tabla "artículos" se deberán realizar dentro de una transacción.

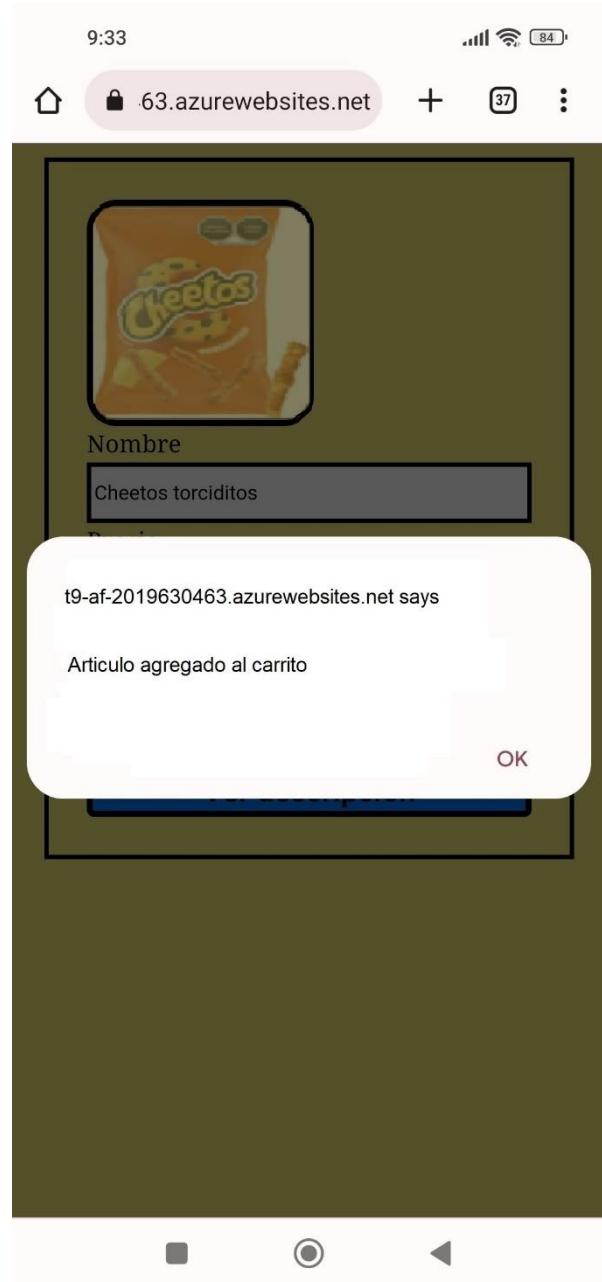
El stock original para este producto es de 4 artículos si se desea comprar la cantidad de 5 artículos aparecerá un mensaje indicando que el numero excede o es menor a 0 y no se podrá comprar hasta elegir una cantidad menor.



Podemos ver que aparece un mensaje en tipo de alerta indicando que no se puede comprar, así que para poder comprar ingresamos una cantidad menor a 4, en este caso 2.



Ingresando cantidad menor al stock podemos ver que se agrega el articulo al carrito.



R7: La pantalla de "Compra de artículos" deberá disponer de un botón "Carrito de compra" el cual deberá desplegar una pantalla "Artículos en el carrito" con la lista de artículos en la tabla "carrito_compra", incluyendo una pequeña imagen del artículo, nombre del artículo, cantidad, precio y costo (cantidad x precio). Así mismo, en la ventana "Artículos en el carrito" se deberá desplegar el total de la compra.

En la pantalla "Compra de artículos" se despliega un botón "Carrito de compras", así que damos clic sobre el para ver el artículo que hemos agregado al carrito este dentro de este.

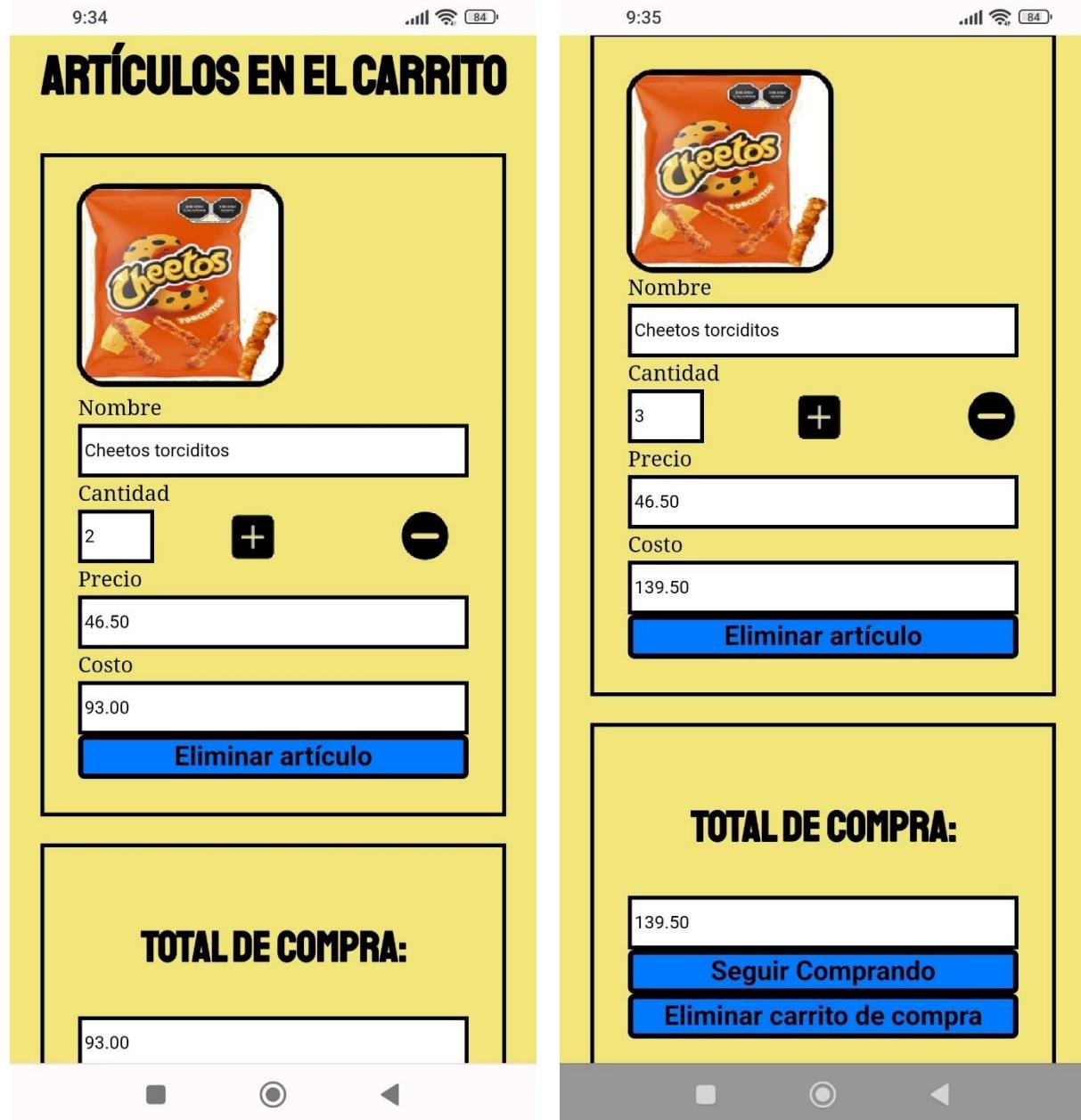


Se despliega la pantalla “Artículos en el carrito” y podemos ver en el carrito de compra, el nombre de los artículos, la cantidad de artículos, el precio, el costo y el total de compra. De igual forma se despliega la opción “Eliminar artículo” (Requerimiento 8).

Al agregar dos artículos podemos ver que el costo es de \$93.00 ya que cada uno tiene un precio de \$46.50



Aquí se puede ver el botón para eliminar el carrito de compra o seguir comprando, de igual forma se puede modificar la cantidad de artículos desde el carrito con los botones "+" y "-". Si observamos en la primera imagen la cantidad de artículos a comprar es de 2, así que presionando el botón "+" incrementamos a 3 artículos, recordemos que el stock es de 4, por lo tanto, no se muestra ningún mensaje ya que no excede el stock (*Requerimiento 9*).



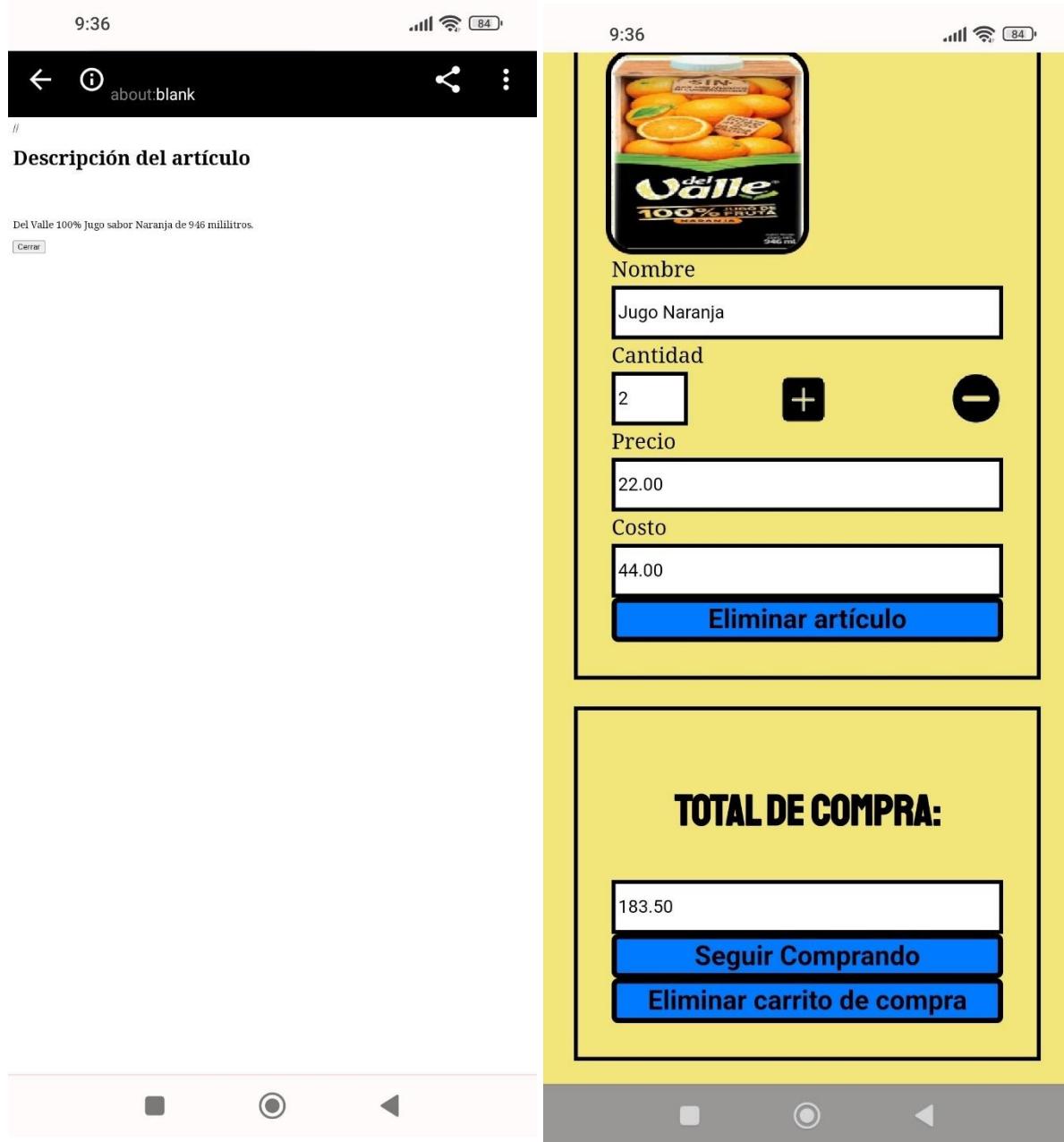
Ahora buscamos, el segundo producto y modificamos la cantidad de 1 a 2 artículos a agregar al carrito, después damos clic en ver descripción.

The image displays two side-by-side screenshots of a mobile application interface, both titled "63.azurewebsites.net".

Screenshot 1 (Left): The title is "COMPRA DE ARTÍCULOS". It shows a search bar with "Jugo Naranja" and "Naranja", a "Buscar artículo" button, and navigation buttons for "Carrito de compras" and "Regresar a principal". Below this, the title "ARTICULOS ENCONTRADOS" is displayed, followed by a product card for "Jugo Naranja" (orange juice) with an image of a carton. Below the card are input fields for "Nombre" (Name), "Precio" (Price), and "Cantidad" (Quantity), each containing the values "Jugo Naranja", "22.00", and "2" respectively.

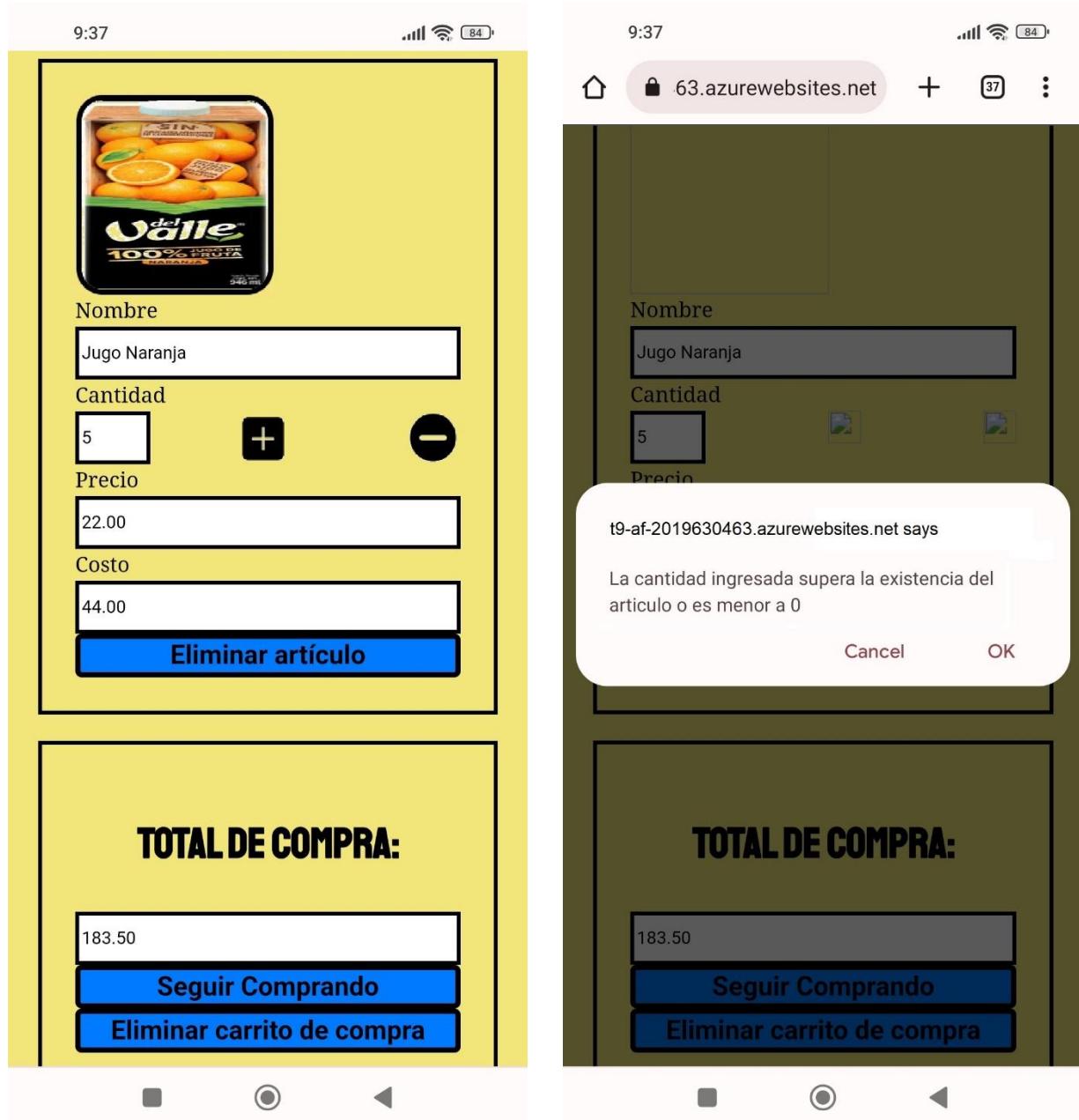
Screenshot 2 (Right): The title is "ARTICULOS ENCONTRADOS". It shows a search bar with "Naranja" and a "Buscar artículo" button, and navigation buttons for "Carrito de compras" and "Regresar a principal". Below this, the title "ARTICULOS ENCONTRADOS" is displayed, followed by a product card for "Jugo Naranja" (orange juice) with an image of a carton. Below the card are input fields for "Nombre" (Name), "Precio" (Price), and "Cantidad" (Quantity), each containing the values "Jugo Naranja", "22.00", and "2" respectively. At the bottom are two blue buttons: "Comprar" (Buy) and "Ver descripción" (View description).

Podemos ver la descripción del artículo, después vamos al carrito de compras de nuevo, para ver si los productos han sido añadidos, vemos que el costo de dos Jugos es de \$44.00 más \$139.50 de los artículos anteriores, suman un total de \$183.50.



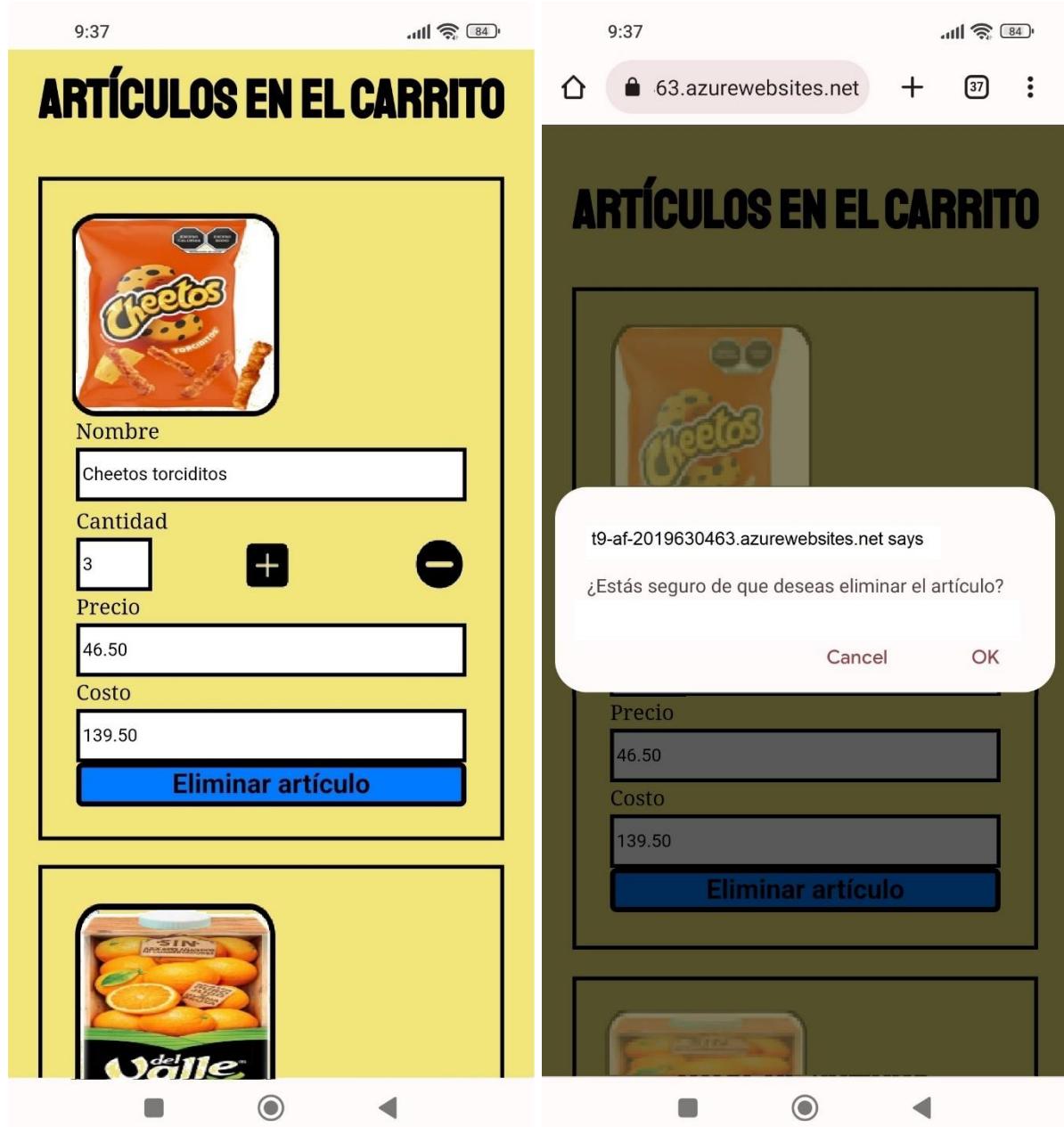
R9: Para cada artículo en la pantalla "Artículos en el carrito" se deberá incluir un control que permita incrementar o decrementar la cantidad del artículo en el carrito. Al incrementar la cantidad se deberá verificar si hay suficientes artículos en la tabla "artículos", si no es así se deberá mostrar una ventana indicando esta situación. Al incrementar o decrementar la cantidad del artículo en el carrito se deberá actualizar (UPDATE) la cantidad en la tabla "artículos" y en la tabla "carrito_compra" dentro de una transacción.

Ahora si incrementamos la cantidad del artículo a 5 desde el carrito de compras usando el botón "+", se desplegará un mensaje de que se supera el stock del artículo ya que el stock de este artículo es igual a 3, entonces se decrementa la cantidad de este artículo a 2, usando el botón "-".



R8: Para cada artículo en la pantalla "Artículos en el carrito" se deberá incluir un botón "Eliminar artículo", al dar clic a este botón se deberá eliminar el artículo del carrito de compra, borrando el registro correspondiente de la tabla "carrito_compra" y agregando la cantidad de los artículos en la tabla "articulos"

Ahora eliminaremos el siguiente artículo del carrito, damos click en "Eliminar artículo" y se desplegará un mensaje preguntando si estas seguro de eliminar el articulo del carrito, damos click en "OK".



Aquí podemos ver que el artículo se ha eliminado correctamente, vemos en el carrito de compras que queda el último artículo que agregamos, y el total de compra queda en \$44.00 ya que solo son dos artículos los que se han agregado al carrito.

The image consists of two side-by-side screenshots of a mobile application interface, both titled "ARTÍCULOS EN EL CARRITO".

Screenshot 1 (Left): Shows a list of items in the cart. One item, a box of Cheetos, has a delete button next to it. A modal dialog box is displayed over the list, containing the message "t9-af-2019630463.azurewebsites.net says Artículo eliminado correctamente" (Article deleted correctly) with "Cancel" and "OK" buttons.

Screenshot 2 (Right): Shows the cart with one item remaining: a container of "Jugo Naranja" (Orange juice). The cart details are as follows:

- Nombre: Jugo Naranja
- Cantidad: 2
- Precio: 22.00
- Costo: 44.00

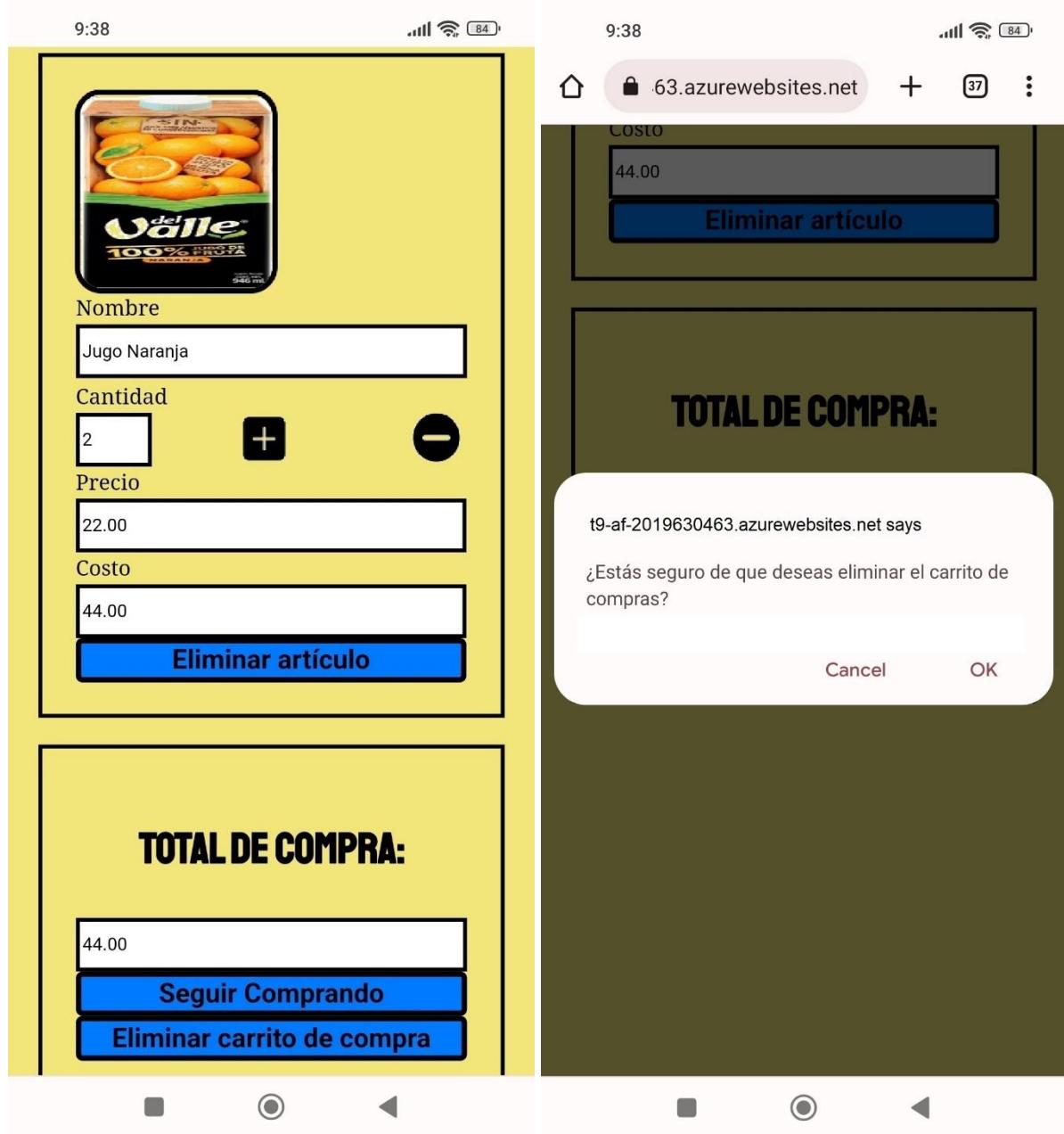
An "Eliminar artículo" (Delete article) button is visible at the bottom of the cart section. Below the cart, a "TOTAL DE COMPRA:" (Total Purchase) section displays the amount 44.00, and a "Seuir Comprando" (Continue Shopping) button is shown.

R10: La pantalla "Artículos en el carrito" deberá tener un botón "Eliminar carrito de compra", al dar clic a este botón se deberá regresar la cantidad de cada artículo a la tabla "articulos" y se deberá borrar todos los registros de la tabla "carrito_compra".

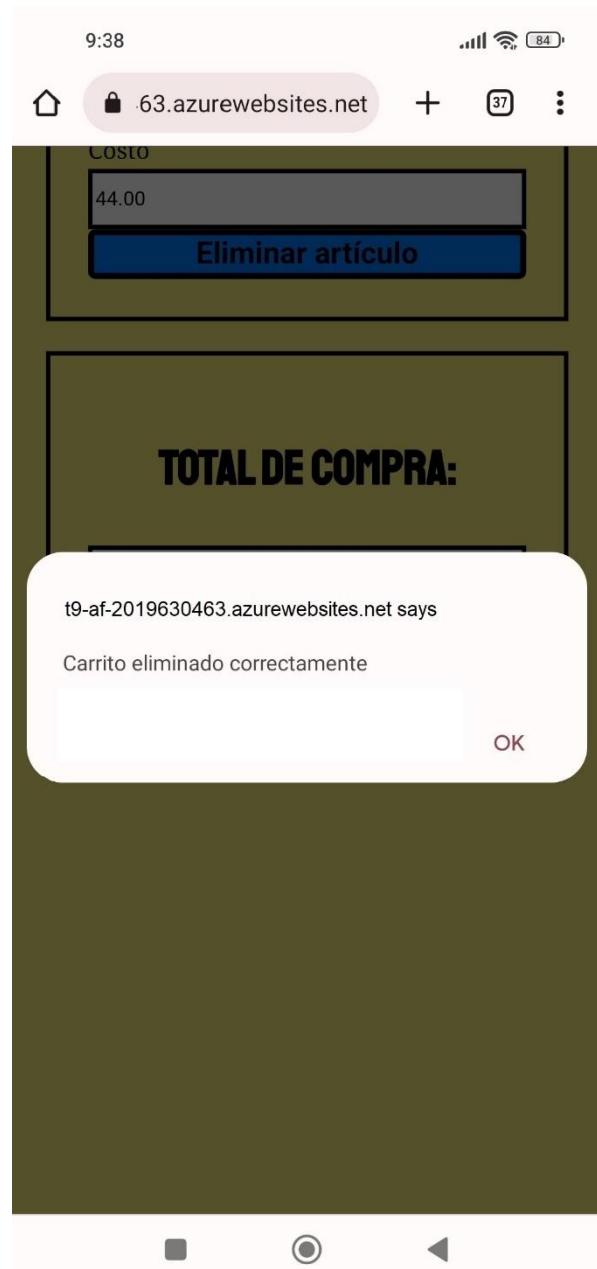
Se deberá desplegar una ventana preguntando al usuario si está seguro de eliminar el carrito de compra, incluyendo dos botones "Si" y "No"; si el usuario presiona el botón "Si" se deberá eliminar el carrito de compra.

Las actualizaciones (UPDATE) de la tabla "articulos" y la eliminación (DELETE) de los registros de la tabla "carrito_compra" deberán realizarse dentro de una transacción

Ahora dentro del carrito de compras daremos click en el botón "Eliminar carrito de compra" y se desplegará un mensaje de si estamos seguros de eliminar el carrito de compras, damos click en "OK"



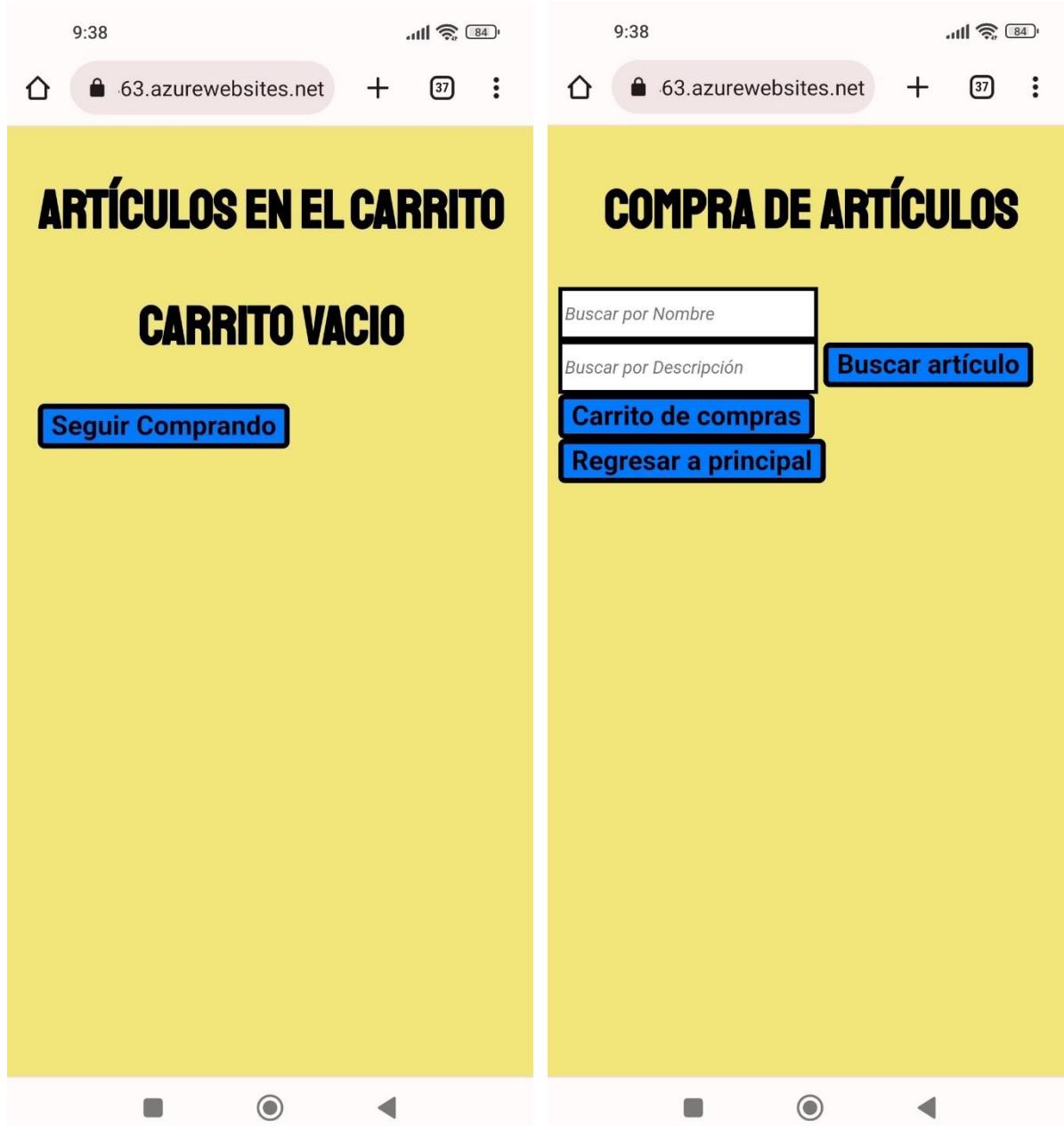
Ahora se despliega un mensaje de que el carrito de compras fue eliminado correctamente y damos click en “OK”.



R11: La pantalla "Artículos en el carrito" deberá tener un botón "Seguir comprando" el cual deberá permitir regresar a la pantalla "Compra de artículos".

Si damos clic en el botón de eliminar carrito primero nos mostrará un mensaje de si se desea eliminar, le damos OK y nos indica que el carrito ha sido eliminado correctamente.

Finalmente podemos ver que el carrito ha quedado vacío, se desplegará un botón para "Seguir comprando", si damos click en el se nos redireccionará a la pantalla "Compra de artículos", así que damos click y aparecer nuevamente las opciones para búsqueda de productos y seguir comprando.



Conversación con Chat GPT.

Flores Castro Luis Antonio.

¿Hola buena noche me podrías ayudar con lo siguiente por favor?

Necesito crear una página web con una estructura básica. En la sección <head>, vamos a incluir algunas etiquetas importantes. Primero, tenemos la etiqueta <meta charset="UTF-8">, que nos permitirá utilizar diferentes caracteres y idiomas en nuestra página web. Luego, encontramos la etiqueta <meta http-equiv="X-UA-Compatible" content="IE=edge">, que asegura la compatibilidad con Internet Explorer. Después, utilizaremos la etiqueta <meta name="viewport" content="width=device-width, initial-scale=1.0"> para que nuestra página se ajuste correctamente en dispositivos móviles. También vamos a agregar un enlace a una fuente externa, que se encuentra en 'https://fonts.googleapis.com/css2?family=Staatliches&display=swap', y la etiqueta <title> con el título 'Tarea 7' para mostrar en la pestaña del navegador.

Dentro del <body>, comenzamos con un contenedor principal identificado por el atributo id="Principal". Este contenedor tiene un estilo en línea definido con style="display: block;", lo que significa que se mostrará en bloque en la página.

Dentro del contenedor principal, encontramos otro elemento <div> con la clase 'titulo'. Dentro de este div, tenemos un encabezado <h1> con el texto 'Articulos'. Esto se mostrará como el título principal de la página.

A continuación, tenemos otro <div> con la clase 'menuPrincipal'. Aquí incluiremos un par de botones. El primer botón tiene un evento onclick="ultima()", que ejecutará una función llamada 'ultima()' cuando se haga clic en él. El segundo botón tiene un evento onclick="mostrarOcultar('Compra_articulos','Principal')", que ejecutará una función llamada 'mostrarOcultar()' pasando los parámetros 'Compra_articulos' y 'Principal'. Esto nos permitirá mostrar u ocultar un elemento con el ID 'Compra_articulos' al hacer clic en el botón.

Chat GPT.

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://fonts.googleapis.com/css2?family=Staatliches&display=swap" rel="stylesheet">
  <title>Tarea 7</title>
</head>
<body>
```

```

<div id="Principal" style="display: block;" >
    <div class="titulo">
        <h1>Articulos</h1>
    </div>
    <!-- MENU PRINCIPAL -->
    <div class="menuPrincipal">
        <button onclick="ultima()">Captura artículo</button>
        <button onclick="mostrarOcultar('Compra_articulos','Principal')">Compra de artículos</button>
    </div>
</div>

```

Flores Castro Luis Antonio.

Después del contenedor principal, encontramos otro `<div>` identificado con el atributo `id="Compra_articulos"`. Este div tiene un estilo en línea definido con `style="display: none;"`, lo que significa que inicialmente estará oculto en la página.

Dentro de este div, tenemos un encabezado de nivel 2 `<h2>` con el texto 'Compra de artículos', que será mostrado como un subtítulo en la página.

A continuación, tenemos un párrafo `<p>` con el texto 'Seleccione los artículos que desea comprar:', que proporciona una breve instrucción al usuario.

Luego, encontramos una serie de elementos `<input type="checkbox">` seguidos de etiquetas `<label>`. Cada `<input>` representa un artículo disponible para comprar y tiene un atributo `id`, `name` y `value` únicos. Los `<label>` están asociados a cada `<input>` a través del atributo `for`, y se utilizan para mostrar el nombre del artículo. En este caso, hay tres artículos: 'Artículo 1', 'Artículo 2' y 'Artículo 3'.

Finalmente, hay un botón `<button>` con el texto 'Realizar compra'. Al hacer clic en este botón, se ejecutará una función llamada 'realizarCompra()

Chat GPT.

```

<div id="Captura_articulos" style="display:none">
    <h1>Captura de artículos</h1>
    <label for="">Nombre artículo</label>
    <input type="text" id="Captura_nombre" placeholder="Nombre">
    <br>
    <label for="">Descripción artículo</label>
    <textarea id="Captura_descripcion" placeholder="Descripción"></textarea>
    <br>

```

```

<label for="">Precio artículo</label>
<input type="text" id="Captura_precio" placeholder="Precio">
<br>
<label for="">Cantidad artículo</label>
<input type="text" id="Captura_cantidad" placeholder="Cantidad" value="4">
<br>
<label for="">Imagen artículo</label>
<input type="file" id="Captura_fotografia">
<br>
<button onclick="ingresarArticulo()">Ingresar artículo</button>
<button onclick="mostrarOcultar('Principal','Compra_articulos')">Regresar a principal</button>
</div>

```

Flores Castro Luis Antonio.

Quiero añadir una sección en la página web para la compra de artículos. Empezaremos con un `<div>` identificado por el atributo `id="Compra_articulos"`. Este div estará oculto inicialmente en la página, para que solo aparezca cuando sea necesario.

Dentro de este div, tendremos un título principal de nivel 1 `<h1>` con el texto 'Compra de artículos'. Esto será el encabezado principal de la sección.

A continuación, crearemos otro div con el atributo `id="Campos_articulos"`. Este div contendrá los campos de búsqueda para encontrar los artículos deseados. Incluirá dos campos de búsqueda `<input type="search">`. El primer campo tendrá el atributo `id="Consulta_nombre"` y un marcador de posición 'Buscar por Nombre'. El segundo campo de búsqueda tendrá el atributo `id="Consulta_descripcion"` y un marcador de posición 'Buscar por Descripción'. Estos campos permitirán a los usuarios buscar artículos según su nombre o descripción. Además, agregaremos un botón `<button>` con el atributo `id="Buscar_articulo"` y el texto 'Buscar artículo'. Al hacer clic en este botón, se ejecutará una función llamada 'buscarArticulo()' para realizar la búsqueda.

Después, crearemos otro div con el atributo `id="Btn-carrito"`. Aquí incluiremos un par de botones. El primer botón tiene el texto 'Carrito de compras' y, al hacer clic en él, ejecutará la función 'mostrarOcultar()' pasando los parámetros 'Articulos_carrito', 'Compra_articulos' y 'Ver_articulo'. Este botón permitirá mostrar u ocultar un elemento con el ID 'Articulos_carrito'. El segundo botón tiene el texto 'Regresar a principal' y, al hacer clic en él, ejecutará la función 'mostrarOcultar()' pasando los parámetros 'Principal', 'Compra_articulos' y 'Ver_articulo'. Este botón permitirá volver al contenido principal de la página.

Chat GPT.

```

<div id="Compra_articulos" style="display: none;">
    <h1>Compra de artículos</h1>
    <div id="Campos_articulos">
        <input type="search" id="Consulta_nombre" placeholder="Buscar por Nombre">

```

```

<input type="search" id="Consulta_descripcion" placeholder="Buscar por Descripción">
<button id="Buscar_articulo" onclick="buscarArticulo()">Buscar artículo</button>

</div>

<div id="Btn-carrito">
    <button onclick="mostrarOcultar('Articulos_carrito','Compra_articulos','Ver_articulo')">Carrito de compras</button>
    <button onclick="mostrarOcultar('Principal','Compra_articulos','Ver_articulo')">Regresar a principal</button>
</div>

</div>

```

Flores Castro Luis Antonio.

Ahora quiero agregar una sección para mostrar los artículos que se encuentran en el carrito de compras. Comenzaremos con un `<div>` identificado por el atributo `id="Articulos_carrito"`. Este div estará oculto inicialmente en la página, para que solo se muestre cuando sea necesario.

Dentro de este div, incluiremos un encabezado principal de nivel 1 `<h1>` con el texto 'Artículos en el carrito'. Esto será el título principal de esta sección.

Luego, crearemos otro div con el atributo `id="Carrito_Vacio"`. Este div también estará oculto inicialmente, y se mostrará solo cuando el carrito esté vacío. Dentro de este div, tendremos otro encabezado principal de nivel 1 `<h1>` con el texto 'Carrito Vacío'. Además, incluiremos un botón `<button>` con el atributo `id="Btn_cerrar"`. Al hacer clic en este botón, se ejecutará la función `'mostrarOcultar()'` pasando los parámetros `'Compra_articulos'`, `'Articulos_carrito'` y `'Ver_articulo'`. Este botón permitirá seguir comprando al ocultar el carrito vacío.

Después, crearemos otro div con el atributo `id="ContenidoArticulos"`. Aquí vamos a mostrar los detalles de los artículos en el carrito. Incluiremos una imagen `` con la ruta de origen `'Articulo2.jpg'`, que representará visualmente el artículo. También incluiremos etiquetas `<label>` para mostrar información sobre el artículo, como el nombre, la cantidad, el precio y el costo. Para cada uno de estos detalles, utilizaremos elementos `<input>` con diferentes tipos de campos (texto, número) y atributos (`readonly`). Estos campos mostrarán los valores correspondientes para el artículo en el carrito.

Además, agregaremos un botón `<button>` con el atributo `id="Eliminar_articulo"`. Al hacer clic en este botón, se ejecutará la función `'eliminarConfirmacion()'`, que permitirá confirmar la eliminación del artículo del carrito.

Chat GPT.

```
<div id="Articulos_carrito" style="display:none;">
    <h1>Artículos en el carrito</h1>

    <div id="Carrito_Vacio" style="display:none;">
        <h1>Carrito Vacío</h1>
        <button id="Btn_cerrar"
            onclick="mostrarOcultar('Compra_articulos','Articulos_carrito','Ver_articulo')">Seguir
            Comprando</button>
    </div>

    <div id="ContenidoArticulos" >
        
        <label for="">Nombre</label>
        <input type="text" id="Consulta_nombre" readonly value="Cheetos torcidos">
        <label for="">Cantidad</label>
        <input type="number" id="Consulta_cantidad" value="2">
        <label for="">Precio</label>
        <input type="text" id="Consulta_precio" readonly value="46.50">
        <label for="">Costo</label>
        <input type="text" id="Costo" readonly value="93.00">

        <button id="Eliminar_articulo" onclick="eliminarConfirmacion()">Eliminar artículo</button>
    <div>
    </div>
</div>
```

Ahora quiero mostrar la información de los artículos en el carrito de compras. Empezaremos con un `<div>` identificado por el atributo `id="ContenidoArticulos"`. Dentro de este div, añadiremos un encabezado de nivel 2 `<h2>` con el texto 'Total de compra'. Después, incluiremos un campo de texto `<input>` con el atributo `type="text"`, `id="Total_compra"`, y el atributo `readonly` para que no se pueda editar. Este campo mostrará el valor total de la compra y, en este caso, tiene un valor predefinido de '93.00'.

A continuación, agregaremos dos botones `<button>`. El primero, con el atributo `id="Btn_cerrar"`, ejecutará la función 'mostrarOcultar()' al hacer clic en él. Esta función pasará los parámetros 'Compra_articulos', 'Articulos_carrito' y 'Ver_articulo', permitiendo volver a la sección de compra de artículos. El segundo botón, con el atributo `id="Eliminar_carrito"`, ejecutará la función 'eliminarCarrito()' al hacer clic en él. Esta función eliminará todo el contenido del carrito de compra.

Después, crearé otro `<div>` con el atributo `id="ContenidoArticulos"`. Dentro de este div, incluiremos la imagen del producto usando el elemento ``, con la ruta de origen 'Articulo3.jpg'. Luego, añadiremos etiquetas `<label>` para mostrar información sobre el artículo, como el nombre, la cantidad, el precio y el costo. Cada uno de estos detalles se representará mediante un campo de texto `<input>` con el atributo `readonly` para que no se pueda editar.

A continuación, incluiremos un botón `<button>` con el atributo `id="Eliminar_articulo"`, que ejecutará la función 'eliminarConfirmacion()' al hacer clic en él. Esta función permitirá confirmar la eliminación del artículo del carrito de compras.

Repetiré el mismo patrón para los siguientes conjuntos de `<div>`, que también tendrán el atributo `id="ContenidoArticulos"`. Estos mostrarán la información de otros artículos en el carrito, junto con el total de compra y los botones para seguir comprando o eliminar el carrito de compra.

Chat GPT.

```
<div id="ContenidoArticulos">

    <h2>Total de compra: </h2>

    <input type="text" id="Total_compra" readonly value="93.00">

    <button id="Btn_cerrar"
    onclick="mostrarOcultar('Compra_articulos','Articulos_carrito','Ver_articulo')"
    Seguir Comprando</button>

    <button id="Eliminar_carrito" onclick=eliminarCarrito()>Eliminar carrito
    de compra</button>

</div>

<!-- Producto 2 ELIMINAR -->

<div id="ContenidoArticulos">

    
    <label for="">Nombre</label>
    <input type="text" id="Consulta_nombre" readonly value="Jugo Naranja">
    <label for="">Cantidad</label>
    <input type="number" id="Consulta_cantidad" value="2">

</div>
```

```

<label for="">Precio</label>
<input type="text" id="Consulta_precio" readonly value="22.00">
<label for="">Costo</label>
<input type="text" id="Costo" readonly value="44.00">

<button id="Eliminar_articulo" onclick="eliminarConfirmacion()">Eliminar
artículo</button>
</div>

<div id="ContenidoArticulos">
<h2>Total de compra: </h2>
<input type="text" id="Total_compra" readonly value="44.00">
<button id="Btn_cerrar"
onclick="mostrarOcultar('Compra_articulos','Articulos_carrito','Ver_articulo')">
Seguir Comprando</button>
<button id="Eliminar_carrito" onclick=eliminarCarrito()>Eliminar carrito
de compra</button>
</div>

<div id="ContenidoArticulos">
<h2>Total de compra: </h2>
<input type="text" id="Total_compra" readonly value="137.00">
<button id="Btn_cerrar"
onclick="mostrarOcultar('Compra_articulos','Articulos_carrito','Ver_articulo')">
Seguir Comprando</button>
<button id="Eliminar_carrito" onclick=eliminarCarrito()>Eliminar carrito
de compra</button>
</div>
</div>

```

Ahora quiero mostrar la información de los artículos en el carrito de compras. Empezaremos con un `<div>` identificado por el atributo `id="ContenidoArticulos"`. Dentro de este div, añadiremos un encabezado de nivel 2 `<h2>` con el texto 'Total de compra'. Después, incluiremos un campo de texto `<input>` con el atributo `type="text"`, `id="Total_compra"`, y el atributo `readonly` para que no se pueda editar. Este campo mostrará el valor total de la compra y, en este caso, tiene un valor predefinido de '93.00'.

A continuación, agregaremos dos botones `<button>`. El primero, con el atributo `id="Btn_cerrar"`, ejecutará la función 'mostrarOcultar()' al hacer clic en él. Esta función pasará los parámetros 'Compra_articulos', 'Articulos_carrito' y 'Ver_articulo', permitiendo volver a la sección de compra de artículos. El segundo botón, con el atributo `id="Eliminar_carrito"`, ejecutará la función 'eliminarCarrito()' al hacer clic en él. Esta función eliminará todo el contenido del carrito de compra.

Después, crearé otro `<div>` con el atributo `id="ContenidoArticulos"`. Dentro de este div, incluiremos la imagen del producto usando el elemento ``, con la ruta de origen 'Articulo3.jpg'. Luego, añadiremos etiquetas `<label>` para mostrar información sobre el artículo, como el nombre, la cantidad, el precio y el costo. Cada uno de estos detalles se representará mediante un campo de texto `<input>` con el atributo `readonly` para que no se pueda editar.

A continuación, incluiremos un botón `<button>` con el atributo `id="Eliminar_articulo"`, que ejecutará la función 'eliminarConfirmacion()' al hacer clic en él. Esta función permitirá confirmar la eliminación del artículo del carrito de compras.

Repetiré el mismo patrón para los siguientes conjuntos de `<div>`, que también tendrán el atributo `id="ContenidoArticulos"`. Estos mostrarán la información de otros artículos en el carrito, junto con el total de compra y los botones para seguir comprando o eliminar el carrito de compra.

Chat GPT.

```
<div id="Ver_articulo" style="display:none;">
    <!-- <div> -->
    <h1>Articulos encontrados</h1>
    <div id="ArticulosEncontrados1" style="display:none; ">
        
        <label for="">Nombre</label>
        <input type="text" id="Consulta_nombreEncontrados" value="Cheetos torciditos" readonly>
        <label for="">Precio</label>
        <input type="text" id="Consulta_precio" value="46.50" readonly>
        <label for="">Cantidad</label>
        <input type="number" id="Consulta_cantidad" value="1">
        <button id="Boton_compra" onclick="ComprarValidacion()">Comprar</button>
        <div id="datos_consulta" style="display:none">
```

```

        <input type="text" name="" id="Consulta_DescripcionEncontrados"
placeholder="aqui" value="Sabritas Cheetos torciditos sabor queso y chile 240
g">

</div>

<!-- VISUALIZACION DE VENTANA CON DESCRIPCION -->

<button id="Boton_descripcion" onclick="ventanaDesc()">Ver
descripción</button>

</div>

```

Flores Castro Luis Antonio.

Ahora vamos a agregar otra sección para mostrar otro artículo encontrado. Empezaremos con un <div> identificado por el atributo id="ArticulosEncontrados2", el cual estará oculto inicialmente.

Dentro de este div, incluiremos una imagen con la ruta de origen 'Articulo3.jpg', que representará visualmente el segundo artículo encontrado.

Luego, añadiremos etiquetas <label> para mostrar información sobre este artículo, como el nombre, el precio y la cantidad. Utilizaremos campos de texto <input> con diferentes tipos de campos (texto, número) y atributos (readonly) para mostrar los valores correspondientes del artículo.

Además, agregaremos un botón <button> con el atributo id="Boton_compra". Al hacer clic en este botón, se ejecutará la función 'ComprarValidacion()', que permitirá realizar la compra del artículo.

También incluiremos otro <div> con el atributo id="datos_consulta", el cual estará oculto inicialmente. Dentro de este div, añadiremos un campo de texto <input> con el atributo type="text" y el atributo id="Consulta_DescripcionEncontrados". Este campo mostrará la descripción del segundo artículo encontrado.

A continuación, incluiremos otro botón <button> con el atributo id="Boton_descripcion". Al hacer clic en este botón, se ejecutará la función 'ventanaDesc()', que permitirá ver la descripción completa del segundo artículo en una ventana.

Chat GPT.

```

<div id="ArticulosEncontrados2" style="display:none;">

    <label for="">Nombre</label>

    <input type="text" id="Consulta_nombreEncontrados" value="Jugo
Naranja" readonly>

    <label for="">Precio</label>

    <input type="text" id="Consulta_precio" value="22.00"readonly>

    <label for="">Cantidad</label>

    <input type="number" id="Consulta_cantidad" value="1">

```

```

<button id="Boton_compra"
onclick="ComprarValidacion()">Comprar</button>

<div id="datos_consulta" style="display:none">
    <input type="text" name="" id="Consulta_DescripcionEncontrados"
placeholder="aqui" value="Del Valle 100% Jugo sabor Naranja de 946 mililitros.">
</div>

<button id="Boton_descripcion" onclick="ventanaDesc()">Ver
descripción</button>

</div>

```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas funciones JavaScript. Empezaremos con la función 'ventanaDesc()' que se encargará de abrir una ventana emergente con la descripción completa de un artículo.

Dentro de la función, obtendremos referencias a los elementos del DOM necesarios, como el botón de Ver descripción y el campo de entrada de la descripción. Luego, agregaremos un evento click al botón para que al hacer clic, se ejecute una serie de acciones.

Dentro del evento, obtendremos el valor de la descripción del artículo. A continuación, crearemos una nueva ventana emergente utilizando el método 'window.open()'. Esta ventana emergente tendrá un tamaño predefinido.

Después, abriremos un nuevo documento en la ventana emergente utilizando el método 'document.open()'. A continuación, agregaremos contenido HTML a la ventana emergente utilizando el método 'document.write()'. Dentro de este contenido, mostraremos el título 'Descripción del artículo' en un estilo de fuente personalizado, seguido del párrafo que contiene la descripción del artículo. También agregaremos un botón para cerrar la ventana emergente.

Finalmente, cerraremos la escritura en el documento de la ventana emergente utilizando el método 'document.close()'.

Además, tenemos la función 'mostrarCarrito()' que se encargará de mostrar u ocultar la pantalla del carrito de compras. Dentro de esta función, obtendremos referencias a los elementos del DOM necesarios, como el botón del carrito, el div de compra de artículos y el div de artículos del carrito. A través de eventos click en los botones correspondientes, cambiaremos la propiedad 'display' de los elementos para mostrar u ocultar la pantalla del carrito de compras según sea necesario.

Por último, tenemos la función 'mostrarOcultar()' que se encargará de mostrar u ocultar elementos del DOM según las instrucciones proporcionadas. Dentro de esta función, obtendremos referencias a los elementos del DOM a través de los identificadores proporcionados como parámetros. A través de condicionales, cambiaremos la propiedad 'display' de los elementos para mostrar u ocultarlos según sea necesario.

Chat GPT.

```

function ventanaDesc()
{
    // Obtener referencia a los elementos del DOM
    const botonDescripcion =
document.getElementById('Boton_descripcion');

    const descripcionInput =
document.getElementById('Consulta_Descripcion');

    // Agregar evento click al botón de Ver descripción
    botonDescripcion.addEventListener('click', () => {
        // Obtener la descripción del artículo
        const descripcion = descripcionInput.value;

        // Crear la ventana emergente
        const ventanaEmergente = window.open('', 'Descripción del
artículo', 'width=400,height=300');

        // Abrir un nuevo documento en la ventana emergente
        ventanaEmergente.document.open();

        // Agregar el contenido HTML a la ventana emergente
        ventanaEmergente.document.write(`

            <h1 style="font-family:'Staatliches', cursive; text-
transform:uppercase;">Descripción del artículo</h1>

            <p>${descripcion}</p>
            <button onclick="window.close()">Cerrar</button>

        `);

        // Cerrar la escritura en el documento de la ventana emergente
        ventanaEmergente.document.close();
    });
}

```

```

}

// Llamar a la función para habilitar la funcionalidad
//<!-- abrirVentanaDescripcion(); -->

// <!-- FUNCION PARA DESPLEGAR PANTALLA DE CARRITO DE COMPRAS -->
function mostrarCarrito() {

    const btnCarrito = document.getElementById('Btn-carrito');
    const compraArticulos = document.getElementById('Compra_articulos');
    const articulosCarrito =
document.getElementById('Articulos_carrito');
    const btnCerrar = document.getElementById('Btn_cerrar');

    btnCarrito.addEventListener('click', () => {
        compraArticulos.style.display = 'none';
        articulosCarrito.style.display = 'block';
    });

    btnCerrar.addEventListener('click', () => {
        articulosCarrito.style.display = 'none';
        compraArticulos.style.display = 'block';
    });
}

function mostrarOcultar(divMostrar,divOcultar,divOcultar2){
    const divM = document.getElementById(divMostrar);
    const divO = document.getElementById(divOcultar);
    const divO2 = document.getElementById(divOcultar2);
    if (divM) {
        divM.style.display = 'block';
    }
}

```

```

if (div0) {
    div0.style.display = 'none';
    div02.style.display = 'none';
}

```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas funciones adicionales en JavaScript. Empezaremos con la función 'eliminarConfirmacion()' que se encargará de mostrar una ventana emergente de confirmación para eliminar un artículo. Dentro de la función, crearemos la ventana emergente utilizando el método 'window.confirm()', pasando como argumento el mensaje de confirmación. A continuación, verificaremos la respuesta del usuario utilizando un condicional. Si el usuario hace clic en 'Aceptar', mostraremos una alerta indicando que el artículo ha sido eliminado correctamente. Si el usuario hace clic en 'Cancelar' o cierra la ventana emergente, no se realizará ninguna acción adicional.

Luego tenemos la función 'eliminarCarrito()' que mostrará una ventana emergente de confirmación para eliminar todo el carrito de compras. Dentro de la función, crearemos la ventana emergente utilizando el método 'window.confirm()', pasando como argumento el mensaje de confirmación. Al igual que en la función anterior, verificaremos la respuesta del usuario utilizando un condicional. Si el usuario hace clic en 'Aceptar', mostraremos una alerta indicando que el carrito ha sido eliminado correctamente. Además, utilizando referencias a los elementos del DOM, ocultaremos los elementos del carrito de compras utilizando el método 'style.display = "none"' y mostraremos el elemento que indica que el carrito está vacío.

A continuación, tenemos la función 'readSingleFile(files, imagen)' que se encargará de leer un archivo de imagen seleccionado por el usuario y mostrarlo en un elemento de imagen. Dentro de la función, obtendremos el archivo seleccionado utilizando el parámetro 'files' y crearemos una instancia de 'FileReader'. A través del evento 'onload' del lector de archivos, asignaremos el resultado de la lectura a la propiedad 'src' del elemento de imagen pasado como parámetro. También extraeremos la parte de la cadena de datos de la imagen base64 y la almacenaremos en una variable llamada 'Fotografia'

Chat GPT.

```

function eliminarConfirmacion() {
    // Crear la ventana emergente

    var ventanaEmergente = window.confirm("¿Estás seguro de que deseas
    eliminar el artículo?");
    //var hidee = document.getElementById(idName);

    // Verificar la respuesta del usuario
    if (ventanaEmergente) {

```

```

        // El usuario hizo clic en "Aceptar"
        alert("Artículo eliminado correctamente");
        //mostrarOcultar('Compra_articulos',hideo,'')
    } else {
        // El usuario hizo clic en "Cancelar" o cerró la ventana emergente
        // No se realiza ninguna acción adicional
    }
}

function eliminarCarrito() {
    // Crear la ventana emergente
    var ventanaEmergente = window.confirm("¿Estás seguro de que deseas
eliminar el carrito de compras?");
    const valor1 = document.getElementById("Articulos_carrito");
    const valor2 = document.getElementById("Carrito_Vacio");
    const elementosCarrito =
document.querySelectorAll('[id="ContenidoArticulos"]');
    //const valor3 = document.getElementById("ContenidoArticulos");
    // Verificar la respuesta del usuario
    if (ventanaEmergente) {
        // El usuario hizo clic en "Aceptar"
        alert("Carrito eliminado correctamente");
        elementosCarrito.forEach(elemento => {
            elemento.style.display = "none";
        });
        valor2.style.display='block';
    } else {
        // El usuario hizo clic en "Cancelar" o cerró la ventana emergente
        // No se realiza ninguna acción adicional
    }
}

```

```

// <!-- FUNCIONES PARA PRODUCIR JSON -->

// <!-- FUNCION PARA LEER FOTOGRAFIA -->
function readSingleFile(files,imagen)
{
    var file = files[0];
    if (!file) return;
    var reader = new FileReader();
    reader.onload = function(e)
    {
        imagen.src = reader.result;
        // reader.result incluye al principio: "data:image/jpeg;base64,"
        Fotografia = reader.result.split(',')[1];
    };
    reader.readAsDataURL(file);
}

```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas funciones adicionales en JavaScript. Comenzaremos con la función 'capturaArticulo()' que se encargará de capturar los datos de un artículo para realizar su alta. Primero, crearemos una instancia del cliente de servicio web (WSClient) utilizando la URL proporcionada. A continuación, declararemos un objeto 'Articulo' y asignaremos los valores de los campos de captura del artículo utilizando los métodos 'getElementById()' para obtener los valores de los elementos del DOM. Finalmente, utilizaremos el método 'postJson()' del cliente de servicio web para enviar los datos del artículo al servidor. Dentro de la función de respuesta, verificaremos el código de respuesta. Si es 200, mostraremos una alerta indicando que el artículo se ha agregado correctamente. De lo contrario, mostraremos el resultado en formato JSON.

Luego tenemos la función 'comprarArticulo()' que realizará la consulta de compra de un artículo. Al igual que antes, crearemos una instancia del cliente de servicio web utilizando la URL. Utilizaremos el método 'postJson()' para enviar una solicitud de compra al servidor, pasando el nombre y la descripción del artículo como parámetros. En la función de respuesta, verificaremos el código de respuesta. Si es 200, asignaremos el valor de la fotografía del resultado a la propiedad 'src' de un elemento de imagen. También asignaremos los valores del nombre y precio del resultado a los elementos correspondientes en el DOM. Si el código de respuesta no es 200, mostraremos una alerta indicando que el artículo buscado no está disponible.

Por último, tenemos la función 'descripcionCarrito()' que se utilizará únicamente para la ventana del carrito de compras. Aquí, crearemos una instancia del cliente de servicio web y declararemos un objeto 'Articulo' que contendrá la descripción del artículo obtenida del elemento del DOM. Sin

embargo, parece haber un error en la declaración del cliente de servicio web, ya que la variable 'URL' no está definida previamente.

Chat GPT.

```
function capturaArticulo(){

    var URL = "/Servicio/rest/ws";
    var cliente = new WSClient(URL);
    // var fotografia = null;

    //Aqui ingresar todos los campos para el alta de un producto.

    var Articulo={

        Nombre: document.getElementById("Captura_nombre").value,
        Descripcion:
        document.getElementById("Captura_descripcion").value,
        Precio: document.getElementById("Captura_precio").value,
        Cantidad: document.getElementById("Captura_cantidad").value,
        Fotografia: Fotografia
    };

    cliente.postJson("CapturaArticulo",
    {
        Articulo: Articulo
    },
    function(code,result)
    {
        if(code==200)
            alert("Articulo agregado correctamente :)");
        else
            alert(JSON.stringify(result));
    });
}

// <!-- FUNCION PARA REALIZAR LA PRIMERA CONSULTA EN LA BUSQUEDA -->
```

```

function comprarArticulo(){

    var URL = "/Servicio/rest/ws";
    var cliente = new WSClient(URL);
    // var fotografia = null;

    cliente.postJson("Compra_articulos",
    {
        Nombre:document.getElementById("Consulta_nombre").value,
        Descripcion:document.getElementById("Consulta_descripcion").value
    },

    function(code,result)
    {
        //Primero se comprueba que exista el producto
        if(code==200){
            Fotografia=result.Fotografia;

            document.getElementById("Consulta_fotografia").src=Fotografia != null ?
            "data:image/jpeg; base64,"+Fotografia: "/usuario_sin_foto.pbg";

            document.getElementById("Consulta_nombre").value=result.Nombre;

            document.getElementById("Consulta_precio").value=result.Descripcion;
            //
            document.getElementById("Producto_precio").value=result.Precio;
            //
            document.getElementById("Producto_cantidad").value=result.Cantidad;

        }else{
            alert("El articulo buscado no se encuentra
disponible"+JSON.stringify(result));
        }
    });
}

```

```

}

//<!-- ESTA CONSULTA SOLO SE HACER PARA LA VENTANA DEL CARRITO DE
COMPRAS -->

function descripcionCarrito(){
    var cliente = new WSClient(URL)
    var Articulo={
        Descripcion:document.getElementById("ConsultaDescripcion").value
    }
}

```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas funciones adicionales en JavaScript. Comenzaremos con la función 'actualizarCarrito()', que se encargará de actualizar el carrito de compras. Primero, crearemos una instancia del cliente de servicio web (WSClient) utilizando la URL proporcionada. A continuación, declararemos un objeto 'Articulo' y asignaremos los valores de los campos de captura del artículo utilizando los métodos 'getElementById()' para obtener los valores de los elementos del DOM. Por último, dentro de la función, podrás realizar la lógica para actualizar el carrito de compras utilizando el objeto 'Articulo'.

Luego tenemos la función 'ComprarValidacion()', que se utilizará para validar una compra. Primero, utilizaremos los métodos 'getElementById()' para obtener los valores de los elementos del DOM 'Consulta_cantidad' y 'Captura_cantidad'. Luego, realizaremos una comparación para verificar si la cantidad ingresada es mayor a la existencia del artículo o menor a cero. Si alguna de estas condiciones se cumple, mostraremos una alerta indicando que la cantidad ingresada no es válida. Si la cantidad es válida (es decir, está dentro del rango), mostraremos una alerta indicando que el artículo se ha añadido al carrito de compras con éxito.

Por último, tenemos la función 'ingresarArticulo()', que se utilizará para indicar que un artículo ha sido agregado exitosamente. Aquí, simplemente mostraremos una alerta indicando que el artículo se ha agregado exitosamente y luego llamaremos a la función 'mostrarOcultar()' para mostrar u ocultar los elementos correspondientes en el DOM.

Chat GPT.

```

function actualizarCarrito(){

    var cliente = new WSClient(URL);

    var Articulo={

        Nombre: document.getElementById("Captura_nombre").value,
        Descripcion: document.getElementById("Captura_descripcion").value,
        Precio: document.getElementById("Captura_precio").value,
        Cantidad: document.getElementById("Captura_cantidad").value,
        Fotografia: Fotografia
    }
}

```

```

    };

}

function ComprarValidacion(){
    const valor = document.getElementById(Consulta_cantidad);
    const valor2 = document.getElementById(Captura_cantidad);
    if(valor>valor2 || valor<1)
        alert("La cantidad ingresada supera la existencia del articulo o es menor a 0");
    if(valor<=valor2 && valor>0)
        alert("Añadido al carrito de compras con éxito")
}

function ingresarArticulo(){
    alert("Artículo agregado exitosamente.")
    mostrarOcultar('Principal','Captura_articulos')
}

```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas funciones adicionales en JavaScript. Comenzaremos con la función 'buscarArticulo()', que se utilizará para buscar un artículo. Primero, utilizaremos los métodos 'getElementById()' para obtener los valores de los elementos del DOM 'Consulta_nombre' y 'Consulta_descripcion'. También obtendremos los valores de los elementos 'Consulta_nombreEncontrados' y 'Consulta_DescripcionEncontrados' que serán utilizados para realizar la comparación. A continuación, utilizaremos condicionales (if-else) para verificar si los valores ingresados coinciden con los criterios de búsqueda establecidos. Si se cumple alguna de las condiciones, utilizaremos los métodos 'getElementById()' para obtener las referencias a los elementos del DOM 'Ver_articulo', 'ArticulosEncontrados1' y 'ArticulosEncontrados2', y modificaremos sus estilos para mostrar u ocultar los elementos correspondientes en función de la búsqueda realizada. En caso de no encontrar ningún producto, se mostrará una alerta indicando que el producto no ha sido encontrado.

Luego, tenemos la función 'ultima()', que se utilizará para realizar la última acción. En esta función, obtendremos las referencias a los elementos del DOM 'Captura_articulos' y 'Principal' utilizando los métodos 'getElementById()'. Luego, modificaremos los estilos de estos elementos para mostrar u ocultar los elementos correspondientes.

Chat GPT.

```
function buscarArticulo(){

    const valor1 = document.getElementById("Consulta_nombre").value;
    const valor2 = document.getElementById("Consulta_descripcion").value;
    const comparacion =
document.getElementById("Consulta_nombreEncontrados").value;

    const comparacion2 =
document.getElementById("Consulta_DescripcionEncontrados").value;
    const div0 = document.getElementById("Ver_articulo");
    const div02 = document.getElementById("ArticulosEncontrados1");
    const div03 = document.getElementById("ArticulosEncontrados2");

    if(valor1 === "Cheetos torciditos" || valor2 === "torciditos"){

        div0.style.display = 'block';
        div02.style.display = 'flex';
        div03.style.display = 'none';

        //mostrarOcultar('ArticulosEncontrados1','','');
    } else if(valor1 === "Jugo Naranja" || valor2 === "Naranja"){

        div0.style.display = 'block';
        div02.style.display = 'none';
        div03.style.display = 'flex';

    }else{
        alert("Producto no encontrado")
    }

}

function ultima(){

    const valor1 = document.getElementById("Captura_articulos");
    const valor2= document.getElementById("Principal");
```

```
valor1.style.display='flex';
valor2.style.display='none';

}
```

Flores Castro Luis Antonio.

Ahora vamos a definir algunas reglas de estilo en CSS. Comenzaremos estableciendo algunas variables de color y fuentes utilizando la pseudo-clase ':root'. A continuación, estableceremos el tamaño de fuente base para el documento HTML utilizando la regla 'html'. Luego, utilizaremos el selector '*' para establecer el modelo de caja en 'border-box' y heredar este comportamiento en todos los elementos del documento.

Después de esto, definiremos algunas reglas de estilo globales. Para el elemento 'body', estableceremos un diseño de columna, centraremos su contenido y asignaremos un color de fondo y tamaño de fuente específicos. Luego, para el elemento 'input', estableceremos un alto, un ancho y un borde. Además, se aplicará un estilo en cursiva para el texto del marcador de posición.

Para las imágenes ('img'), estableceremos un ancho y alto fijo. Para los botones ('button'), estableceremos propiedades relacionadas con el tamaño, el color de fondo, los bordes y el estilo al pasar el cursor sobre ellos.

A continuación, definiremos algunas reglas de estilo específicas para los elementos con identificadores particulares. Por ejemplo, para el elemento con el ID 'Principal', estableceremos propiedades para centrar su contenido. Para el elemento con el ID 'menuPrincipal', también lo centraremos.

Además, definiremos estilos para los encabezados ('h1', 'h2', 'h3'), estableciendo alineación, color y tipo de fuente específicos.

Para el elemento con el ID 'Consulta_fotografia', aplicaremos un borde y un radio de borde específicos.

También definiremos estilos para los elementos con los ID 'Ver_articulo', 'ContenidoArticulos', 'ArticulosEncontrados1', 'ArticulosEncontrados2' y 'Captura_articulos'. Estableceremos propiedades como relleno, margen, dirección de flexión, alineación y bordes.

Chat GPT.

```
<style>
:root{
```

```

--primario:#F2E579;
--primarioalt:#E5D448;
--secundario:#000;
--blanco:#FFF;
--fuenteprincipal:'Staatliches', cursive;
--morado:#8946FF;

}

html {
    box-sizing: border-box;
    font-size: 62.5%;
}

*, *:before, *:after {
    box-sizing: inherit;
}

/**Globales**/

body{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    background-color: var(--primario);
    font-size: 1.6rem;
    line-height: 1.5;
}

input {
    height: 4rem;
    width: auto;
    border: 3px solid black;
}

```

```
}

input::placeholder {
    font-style: italic;
}

img{
    width:150px;
    height:150px;
}

button {
    width: auto;
    height: auto;
    border-radius: 5px;
    font-family: 'Roboto', sans-serif;
    font-size: 20px;
    font-weight: bold;
    color: black;
    background-color: #007bff;
    border: 4px solid black;
}

button:hover {
    background-color: lightblue;
    cursor: pointer;
}

#Principal{
    display: flex;
    justify-content: center;
```

```
    align-items: center;

}

#menuPrincipal{
    display: flex;
    justify-content: center;
    align-items: center;
}

h1,h2,h3{
    text-align: center;
    color: var(--secundario);
    font-family: 'Staatliches', cursive;
}

h1{
    font-size: 4rem;
}

h2{
    font-size: 3.2rem;
}

h3{
    font-size: 2.4rem;
}

#Consulta_fotografia {
    border: 4px solid black;
    border-radius: 2rem;
}
```

```
#Ver_articulo{
    display: block;
    justify-content: column;
    justify-content: center;
}

#ContenidoArticulos{
    padding: 2.5rem;
    margin-bottom: 2rem;
    display: flex;
    flex-direction: column;
    justify-content: center;
    border: 3px solid black;
}

#ArticulosEncontrados1{
    padding: 2.5rem;
    margin-bottom: 2rem;
    display: flex;
    flex-direction: column;
    justify-content: center;
    border: 3px solid black;
}

#ArticulosEncontrados2{
    padding: 2.5rem;
    margin-bottom: 2rem;
    display: flex;
    flex-direction: column;
    justify-content: center;
```

```
border: 3px solid black;

}

#Captura_articulos{
    padding: 2.5rem;
    margin-bottom: 2rem;
    display: flex;
    flex-direction: column;
    justify-content: center;
    border: 3px solid black;
}


```

</style>

Flores Castro Luis Antonio.

Puedes ayudarme con lo siguiente porfavor, requiero que implementes lo siguiente en C#

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "CapturaArticulos".

Dentro de la clase "CapturaArticulos", crea una clase pública llamada "Articulo" con las siguientes propiedades:

Nombre: cadena de texto

Descripcion: cadena de texto

Precio: número decimal

Cantidad: número entero

Fotografia: arreglo de bytes

Crea otra clase pública llamada "ParametroCapturaArticulo" con una propiedad "articulo" de tipo "Articulo".

Define una clase pública llamada "Error" con una propiedad "mensaje" de tipo cadena de texto.

Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Captura_articulos". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Captura_articulos".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroCapturaArticulo" y almacénalo en una variable llamada "crearArt".

Extrae el objeto "Articulo" de "crearArt" y guárdalo en una variable llamada "articulo".

Realiza las siguientes validaciones:

Verifica si "articulo.Nombre" está vacío o nulo. Si es así, devuelve un resultado de error con el mensaje "El nombre del artículo es obligatorio.".

Verifica si "articulo.Descripcion" está vacío o nulo. Si es así, devuelve un resultado de error con el mensaje "La descripción del artículo es obligatoria.".

Verifica si "articulo.Precio" es menor o igual a cero. Si es así, devuelve un resultado de error con el mensaje "El Precio del artículo es obligatorio o tiene que ser mayor a 0.".

Verifica si "articulo.Cantidad" es menor o igual a cero. Si es así, devuelve un resultado de error con el mensaje "La cantidad del artículo debe ser mayor a 0.".

Verifica si "articulo.Fotografia" es nulo. Si es así, devuelve un resultado de error con el mensaje "Ingrese una imagen del artículo.".

Establece una cadena de conexión obtenida de una variable de entorno llamada "SQLConnectionString".

Crea una nueva conexión SqlConnection utilizando la cadena de conexión y guárdala en una variable llamada "connection".

Abre la conexión llamando al método "Open" en "connection".

Inicia una transacción llamando al método "BeginTransaction" en "connection" y guárdala en una variable llamada "transaction".

Dentro de un bloque "try", realiza lo siguiente:

Crea un nuevo comando SqlCommand llamado "insertarArticulo" con la siguiente instrucción SQL: "INSERT INTO articulos(ID, Nombre, Descripcion, Precio, Cantidad, Fotografia) VALUES (0, @Nombre, @Descripcion, @Precio, @Cantidad, @Fotografia)".

Asigna los valores correspondientes a los parámetros del comando "insertarArticulo" utilizando los valores de "articulo".

Ejecuta el comando "insertarArticulo" llamando al método "ExecuteNonQuery".

Fuera del bloque "try", llama al método "Commit" en "transaction" para confirmar la transacción.

Captura cualquier excepción en un bloque "catch" y haz lo siguiente:

Llama al método "Rollback" en "transaction" para revertir la transacción.

Devuelve un resultado de error con el mensaje de excepción.

Finalmente, fuera del bloque "try-catch", cierra la conexión llamando al método "Close" en "connection".

Devuelve un resultado exitoso utilizando "OkResult".

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error con el mensaje de excepción.

Chat GPT.

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Net;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class CapturaArticulos
    {
        public class Articulo
        {
            public string Nombre { get; set; }
            public string Descripcion { get; set; }
            public double Precio { get; set; }
            public int Cantidad { get; set; }
            public byte[] Fotografia { get; set; }
        }
    }
}
```

```

public class ParametroCapturaArticulo
{
    public Articulo articulo { get; set; }
}

public class Error
{
    public string mensaje { get; set; }

    public Error(string mensaje)
    {
        this.mensaje = mensaje;
    }
}

[FunctionName("Captura_articulos")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Captura_articulos")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParametroCapturaArticulo crearArt =
JsonConvert.DeserializeObject<ParametroCapturaArticulo>(requestBody);
        Articulo articulo = crearArt.articulo;

        // Se realizan las validaciones de inputs como campos
        obligatorios
        if (string.IsNullOrEmpty(articulo.Nombre))
        {

```

```

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("El nombre del
artículo es obligatorio.")));

    }

    if (string.IsNullOrEmpty(articulo.Descripcion))
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("La descripción del
artículo es obligatoria.")));

    }

    if (articulo.Precio <= 0.0)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("El Precio del
artículo es obligatorio o tiene que ser mayor a 0.")));

    }

    if (articulo.Cantidad <= 0)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("La cantidad del
artículo debe ser mayor a 0")));

    }

    if (articulo.Fotografia == null)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("Ingrese una imagen
del artículo.")));

    }

    // Se realiza la conexión

    string connectionString =
Environment.GetEnvironmentVariable("SQLConnectionString");

```

```

        using (SqlConnection connection = new
SqlConnection(connectionString))
{
    connection.Open();
    SqlTransaction transaction = connection.BeginTransaction();

    try
    {
        // Se realiza el INSERT de los datos a la tabla
"articulos"

        using (SqlCommand insertarArticulo = new SqlCommand(
            "INSERT INTO articulos(ID, Nombre, Descripcion,
Precio, Cantidad, Fotografia) VALUES (0, @Nombre, @Descripcion, @Precio,
@Cantidad, @Fotografia)",
            connection,
            transaction))

        {
            insertarArticulo.Parameters.AddWithValue("@Nombre",
articulo.Nombre);

            insertarArticulo.Parameters.AddWithValue("@Descripcion", articulo.Descripcion);
            insertarArticulo.Parameters.AddWithValue("@Precio",
articulo.Precio);

            insertarArticulo.Parameters.AddWithValue("@Cantidad", articulo.Cantidad);

            insertarArticulo.Parameters.AddWithValue("@Fotografia", articulo.Fotografia);
            insertarArticulo.ExecuteNonQuery();
        }

        // Transacción realizada
        transaction.Commit();
    }
    catch (Exception e)
    {

```

```

        transaction.Rollback();

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));

    }

    finally

    {

        connection.Close();

    }

}

return new OkResult();
}

catch (Exception e)

{

    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));

}

}

}

}

}

```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "Borrar_articulo".

Dentro de la clase "Borrar_articulo", crea una clase pública llamada "ParametroBorrarArticulo" con las siguientes propiedades:

ID: cadena de texto

Cantidad: cadena de texto

Crea otra clase pública llamada "Error" con una propiedad "Mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Articulos_carrito". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Articulos_carrito".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroBorrarArticulo" y almacénalo en una variable llamada "eliminarArticulo".

Extrae los valores de "eliminarArticulo.ID" y "eliminarArticulo.Cantidad" y guárdalos en las variables "ID" y "Cantidad" respectivamente.

Crea una nueva conexión SqlConnection utilizando una cadena de conexión específica y guárdala en una variable llamada "connection".

Abre la conexión llamando al método "Open" en "connection".

Dentro de un bloque "using", crea un nuevo comando SqlCommand llamado "command" con la siguiente instrucción SQL: "UPDATE articulos SET Cantidad = Cantidad + (SELECT Cantidad FROM carrito_compra WHERE ID = @ID) WHERE ID = @ID". Asocia el comando a la conexión "connection".

Agrega parámetros al comando "command" utilizando el método "Parameters.AddWithValue". Establece el valor del parámetro "@ID" con la variable "ID".

Ejecuta el comando llamando al método "ExecuteNonQuery" en "command".

Dentro de otro bloque "using", crea otro comando SqlCommand llamado "command" con la siguiente instrucción SQL: "DELETE FROM carrito_compra WHERE ID = @ID". Asocia el comando a la conexión "connection".

Agrega un parámetro al comando "command" utilizando el método "Parameters.AddWithValue". Establece el valor del parámetro "@ID" con la variable "ID".

Ejecuta el comando llamando al método "ExecuteNonQuery" en "command".

Fuera del bloque "using", la conexión se cerrará automáticamente.

Devuelve un resultado exitoso utilizando "OkResult".

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error utilizando "BadRequestObjectResult". Serializa un objeto "Error" con el mensaje de excepción y conviértelo a formato JSON.

Chat GPT.

```
using System;
using System.Data.SqlClient;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class Borrar_articulo
    {
        public class ParametroBorrarArticulo
        {
            public string ID { get; set; }
            public string Cantidad { get; set; }
        }

        public class Error
        {
            public string Mensaje { get; set; }

            public Error(string mensaje)
            {
                Mensaje = mensaje;
            }
        }
    }
}

[FunctionName("Articulos_carrito")]
```

```

public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Articulos_carrito")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParametroBorrarArticulo eliminarArticulo =
JsonConvert.DeserializeObject<ParametroBorrarArticulo>(requestBody);
        string ID = eliminarArticulo.ID;
        string Cantidad = eliminarArticulo.Cantidad;

        using (SqlConnection connection = new
SqlConnection("<connection-string>"))
        {

            connection.Open();

            using (SqlCommand command = new SqlCommand("UPDATE articulos
SET Cantidad = Cantidad + (SELECT Cantidad FROM carrito_compra WHERE ID = @ID)
WHERE ID = @ID", connection))
            {

                command.Parameters.AddWithValue("@ID", ID);
                command.ExecuteNonQuery();
            }

            using (SqlCommand command = new SqlCommand("DELETE FROM
carrito_compra WHERE ID = @ID", connection))
            {

                command.Parameters.AddWithValue("@ID", ID);
                command.ExecuteNonQuery();
            }
        }
    }
}

```

```
        return new OkResult();

    }

    catch (Exception e)
    {
        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
    }
}
```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "ArticulosCarrito".

Dentro de la clase "ArticulosCarrito", crea una clase pública llamada "ParametroConsultaArticulo" con las siguientes propiedades:

ID: entero

Nombre: cadena de texto

Descripción: cadena de texto

Precio: flotante

Fotografia: matriz de bytes

Crea otra clase pública llamada "CarritoCompra" con las siguientes propiedades:

IDCarrito: entero

Cantidad: entero

ID: entero

Nombre: cadena de texto

Descripción: cadena de texto

Precio: flotante

Existencia: entero

Imagen: matriz de bytes

Crea otra clase pública llamada "Error" con una propiedad "Mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Articulos_carrito". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Articulos_carrito".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroConsultaArticulo" y almacénalo en una variable llamada "carritoArticulo".

Extrae los valores de las propiedades de "carritoArticulo" y guárdalos en las variables correspondientes.

Crea una lista de objetos "CarritoCompra" llamada "listaCarrito".

Declara una variable booleana "dataAvailable" y asigna el valor false.

Crea una nueva conexión SqlConnection utilizando una cadena de conexión específica y guárdala en una variable llamada "connection".

Abre la conexión llamando al método "Open" en "connection".

Dentro de un bloque "using", crea un nuevo comando SqlCommand llamado "command" con la siguiente instrucción SQL:

```
"SELECT b.IDCarrito, b.Cantidad, a.ID, a.Nombre, a.Descripcion, a.Precio, a.Existencia, a.Imagen  
FROM articulos a INNER JOIN carrito_compra b ON b.ID = a.ID LEFT OUTER JOIN Fotografia b  
ON a.ID = b.ID".
```

Asocia el comando a la conexión "connection".

Dentro de otro bloque "using", crea un SqlDataReader llamado "reader" utilizando el método "ExecuteReader" en "command".

Dentro de un bucle "while" que itera mientras haya filas disponibles en "reader", realiza lo siguiente:

Establece "dataAvailable" como true.

Crea un nuevo objeto "CarritoCompra" llamado "item".

Asigna los valores de las columnas del lector a las propiedades correspondientes de "item".

Agrega "item" a la lista "listaCarrito".

Después de salir del bucle "while", cierra el lector "reader".

Fuera del bloque "using" del lector, cierra la conexión llamando al método "Close" en "connection".

Comprueba si "dataAvailable" es true.

Si es true, devuelve un resultado exitoso (OkResult) con la lista de carrito convertida a JSON utilizando JsonConvert.SerializeObject().

Si es false, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "El artículo no está disponible" convertido a JSON.

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de excepción convertido a JSON.

Chat GPT

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class ArticulosCarrito
    {
        public class ParametroConsultaArticulo
        {
            public int ID { get; set; }
            public string Nombre { get; set; }
            public string Descripcion { get; set; }
            public float Precio { get; set; }
            public byte[] Fotografia { get; set; }
        }

        public class CarritoCompra
        {
            public int IDCarrito { get; set; }
            public int Cantidad { get; set; }
        }
    }
}
```

```

        public int ID { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public float Precio { get; set; }
        public int Existencia { get; set; }
        public byte[] Imagen { get; set; }
    }

    public class Error
    {
        public string Mensaje { get; set; }

        public Error(string mensaje)
        {
            Mensaje = mensaje;
        }
    }

    [FunctionName("Articulos_carrito")]
    public static IActionResult Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Articulos_carrito")] HttpRequest req,
        ILogger log)
    {
        try
        {
            string requestBody = new StreamReader(req.Body).ReadToEnd();
            ParametroConsultaArticulo carritoArticulo =
JsonConvert.DeserializeObject<ParametroConsultaArticulo>(requestBody);

            int IDarticulo = carritoArticulo.ID;
            string Nombre = carritoArticulo.Nombre;
            string Descripcion = carritoArticulo.Descripcion;
        }
    }
}

```

```

        float Precio = carritoArticulo.Precio;
        byte[] Fotografia = carritoArticulo.Fotografia;

        List<CarritoCompra> listaCarrito = new List<CarritoCompra>();
        bool dataAvailable = false;

        using (SqlConnection connection = new
SqlConnection("<connection-string>"))
{
    connection.Open();

    using (SqlCommand command = new SqlCommand("SELECT
b.IDCarrito, b.Cantidad, a.ID, a.Nombre, a.Descripcion, a.Precio, a.Existencia,
a.Imagen FROM articulos a INNER JOIN carrito_compra b ON b.ID = a.ID LEFT OUTER
JOIN Fotografia b ON a.ID = b.ID", connection))
    {
        using (SqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                dataAvailable = true;

                CarritoCompra item = new CarritoCompra
                {
                    IDCarrito = reader.GetInt32(0),
                    Cantidad = reader.GetInt32(1),
                    ID = reader.GetInt32(2),
                    Nombre = reader.GetString(3),
                    Descripcion = reader.GetString(4),
                    Precio = reader.GetFloat(5),
                    Existencia = reader.GetInt32(6),
                    Imagen = (byte[])reader.GetValue(7)
                };
            }
        }
    }
}

```

```

        listaCarrito.Add(item);
    }
}

}

if (dataAvailable)
{
    return new
OkObjectResult(JsonConvert.SerializeObject(listaCarrito));
}
else
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("El artículo no
está disponible")));
}
}

catch (Exception e)
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
}
}

}
}
}
```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "CapturaArticulos".

Dentro de la clase "CapturaArticulos", crea una clase pública llamada "Articulo" con las siguientes propiedades:

Nombre: cadena de texto

Descripcion: cadena de texto

Precio: doble

Cantidad: entero

Fotografia: matriz de bytes

Crea otra clase pública llamada "ParametroCapturaArticulo" con una propiedad "articulo" de tipo "Articulo".

Crea otra clase pública llamada "Error" con una propiedad "mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Captura_articulos". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Captura_articulos".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroCapturaArticulo" y almacénalo en una variable llamada "crearArt".

Extrae el objeto "articulo" de "crearArt" y guárdalo en una variable llamada "articulo".

Realiza las validaciones necesarias en los campos del artículo:

Verifica si el campo "Nombre" del artículo es nulo o vacío. Si es así, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "El nombre del artículo es obligatorio" convertido a JSON.

Verifica si el campo "Descripcion" del artículo es nulo o vacío. Si es así, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "La descripción del artículo es obligatoria" convertido a JSON.

Verifica si el campo "Precio" del artículo es menor o igual a cero. Si es así, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "El precio del artículo es obligatorio o debe ser mayor a cero" convertido a JSON.

Verifica si el campo "Cantidad" del artículo es menor o igual a cero. Si es así, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "La cantidad del artículo debe ser mayor a cero" convertido a JSON.

Verifica si el campo "Fotografia" del artículo es nulo. Si es así, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje "Ingrese una imagen del artículo" convertido a JSON.

Realiza las operaciones deseadas dentro de la función.

Devuelve un resultado exitoso (OkResult) para indicar que la operación se realizó correctamente.

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de excepción convertido a JSON.

Chat GPT.

```
using System;
using System.IO;
using System.Net;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class CapturaArticulos
    {
        public class Articulo
        {
            public string Nombre { get; set; }
            public string Descripcion { get; set; }
            public double Precio { get; set; }
            public int Cantidad { get; set; }
            public byte[] Fotografia { get; set; }
        }

        public class ParametroCapturaArticulo
```

```

{
    public Articulo articulo { get; set; }
}

public class Error
{
    public string mensaje { get; set; }

    public Error(string mensaje)
    {
        this.mensaje = mensaje;
    }
}

[FunctionName("Captura_articulos")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Captura_articulos")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParametroCapturaArticulo crearArt =
JsonConvert.DeserializeObject<ParametroCapturaArticulo>(requestBody);
        Articulo articulo = crearArt.articulo;

        // Se realizan las validaciones de inputs como campos
        obligatorios
        if (string.IsNullOrEmpty(articulo.Nombre))
        {

```

```

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("El nombre del
artículo es obligatorio.")));

    }

    if (string.IsNullOrEmpty(articulo.Descripcion))
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("La descripción del
artículo es obligatoria.")));

    }

    if (articulo.Precio <= 0.0)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("El Precio del
artículo es obligatorio o tiene que ser mayor a 0.")));

    }

    if (articulo.Cantidad <= 0)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("La cantidad del
artículo debe ser mayor a 0")));

    }

    if (articulo.Fotografia == null)
    {

        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("Ingrese una imagen
del artículo.")));

    }

    // Realizar las operaciones deseadas aquí

```

```
        return new OkResult();

    }

    catch (Exception e)
    {
        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
    }
}
```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "ActualizarArticulo".

Dentro de la clase "ActualizarArticulo", crea una clase pública llamada "ParametroComprarArticulo" con las siguientes propiedades:

ID: entero

Cantidad: entero

Crea otra clase pública llamada "Error" con una propiedad "mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "actualizar_articulo". Este método toma los siguientes parámetros:

Un objeto `HttpRequest` llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "actualizar articulo".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroComprarArticulo" y almacénalo en una variable llamada "comprarArticulo".

Extrae las propiedades "ID" y "Cantidad" de "comprarArticulo" y guárdalas en las variables "IDarticulo" y "cantidad", respectivamente.

Crea una nueva conexión a SQL Server utilizando la cadena de conexión especificada.

Abre la conexión.

Inicia una transacción utilizando el objeto de conexión.

Dentro de un bloque "try", ejecuta las siguientes operaciones dentro de la transacción:

Crea un comando SQL para actualizar el campo "Cantidad" de la tabla "articulos" utilizando los parámetros "@Cantidad" y "@ID".

Ejecuta el comando SQL para actualizar el registro en la tabla "articulos".

Crea otro comando SQL para insertar un nuevo registro en la tabla "carrito_compra" utilizando los parámetros "@ID" y "@Cantidad"

Ejecuta el comando SQL para insertar el registro en la tabla "carrito_compra".

Confirma la transacción.

Dentro del bloque "catch" del bloque "try", captura cualquier excepción y realiza lo siguiente:

Revierte (rollback) la transacción.

Devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de excepción convertido a JSON.

Cierra la conexión.

Devuelve un resultado exitoso (OkResult) para indicar que la operación se realizó correctamente.

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de excepción convertido a JSON.

Chat GPT.

```
using System;
using System.Data.SqlClient;
using System.IO;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class ActualizarArticulo
    {
        public class ParametroComprarArticulo
        {
            public int ID { get; set; }
```

```

        public int Cantidad { get; set; }

    }

    public class Error
    {
        public string mensaje { get; set; }

        public Error(string mensaje)
        {
            this.mensaje = mensaje;
        }
    }

    [FunctionName("actualizar_articulo")]
    public static IActionResult Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"actualizar_articulo")] HttpRequest req,
        ILogger log)
    {
        try
        {
            string requestBody = new StreamReader(req.Body).ReadToEnd();
            ParametroComprarArticulo comprarArticulo =
JsonConvert.DeserializeObject<ParametroComprarArticulo>(requestBody);
            int IDarticulo = comprarArticulo.ID;
            int cantidad = comprarArticulo.Cantidad;

            using (SqlConnection connection = new
SqlConnection("<connection-string>"))
            {
                connection.Open();

                SqlTransaction transaction = connection.BeginTransaction();

```

```

        try
        {
            using (SqlCommand command = new SqlCommand("UPDATE
articulos SET Cantidad=@Cantidad WHERE ID=@ID", connection, transaction))
            {

                command.Parameters.AddWithValue("@Cantidad",
cantidad);
                command.Parameters.AddWithValue("@ID", IDarticulo);

                command.ExecuteNonQuery();
            }

            using (SqlCommand command2 = new SqlCommand("INSERT INTO
carrito_compra (ID, Cantidad) VALUES (@ID, @Cantidad)", connection,
transaction))
            {

                command2.Parameters.AddWithValue("@ID", IDarticulo);
                command2.Parameters.AddWithValue("@Cantidad",
cantidad);

                command2.ExecuteNonQuery();
            }

            transaction.Commit();
        }
        catch (Exception e)
        {
            transaction.Rollback();

            return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
        }
    }

```

```

        return new OkResult();
    }

    catch (Exception e)
    {
        return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
    }
}
}

```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "ConsultaArticulos".

Dentro de la clase "ConsultaArticulos", crea una clase pública llamada "ParamConsultaArticulos" con las siguientes propiedades:

id_articulo: entero

nombre_articulo: cadena de texto

desc_articulo: cadena de texto

precio: flotante

foto: matriz de bytes

Crea otra clase pública llamada "Articulo" con las siguientes propiedades:

id_articulo: entero

nombre_articulo: cadena de texto

desc_articulo: cadena de texto

precio: flotante

cantidad: entero

foto: matriz de bytes

Crea otra clase pública llamada "Error" con una propiedad "Mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "consultaArticulos". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "articulos".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParamConsultaArticulos" y almacénalo en una variable llamada "p".

Extrae las propiedades de "p" y guárdalas en las variables correspondientes.

Crea una lista de objetos "Articulo" llamada "articulos" y una variable booleana "dataAvailable" establecida en falso.

Crea una nueva conexión a SQL Server utilizando la cadena de conexión especificada.

Abre la conexión.

Crea un comando SQL para seleccionar los registros de la tabla "articulos".

Ejecuta el comando SQL utilizando un SqlDataReader y recorre los resultados.

Dentro del bucle, crea un objeto "Articulo", asigna los valores leídos del lector de datos y añade el objeto a la lista "articulos".

Cierra el lector de datos.

Si se encontraron datos disponibles, devuelve un resultado exitoso (OkObjectResult) que contiene la lista de artículos convertida a JSON.

De lo contrario, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene un mensaje indicando que no hay artículos disponibles.

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de la excepción.

Chat GPT.

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
```

```
{  
    public static class ConsultaArticulos  
    {  
        public class ParamConsultaArticulos  
        {  
            public int id_articulo { get; set; }  
            public string nombre_articulo { get; set; }  
            public string desc_articulo { get; set; }  
            public float precio { get; set; }  
            public byte[] foto { get; set; }  
        }  
  
        public class Articulo  
        {  
            public int id_articulo { get; set; }  
            public string nombre_articulo { get; set; }  
            public string desc_articulo { get; set; }  
            public float precio { get; set; }  
            public int cantidad { get; set; }  
            public byte[] foto { get; set; }  
        }  
  
        public class Error  
        {  
            public string Mensaje { get; set; }  
  
            public Error(string mensaje)  
            {  
                Mensaje = mensaje;  
            }  
        }  
    }  
}
```

```

[FunctionName("consultaArticulos")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"articulos")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParamConsultaArticulos p =
JsonConvert.DeserializeObject<ParamConsultaArticulos>(requestBody);
        int id_articulo = p.id_articulo;
        string nombre_articulo = p.nombre_articulo;
        string desc_articulo = p.desc_articulo;
        float precio = p.precio;
        byte[] foto = p.foto;

        List<Articulo> articulos = new List<Articulo>();
        bool dataAvailable = false;

        using (SqlConnection connection = new
SqlConnection("<connection-string>"))
        {
            connection.Open();

            using (SqlCommand command = new SqlCommand("SELECT
id_articulo, nombre_articulo, desc_articulo, precio, cantidad, foto FROM
articulos", connection))
            {
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())

```

```

    {

        dataAvailable = true;

        Articulo a = new Articulo();

        a.id_articulo = reader.GetInt32(0);

        a.nombre_articulo = reader.GetString(1);

        a.desc_articulo = reader.GetString(2);

        a.precio = reader.GetFloat(3);

        a.cantidad = reader.GetInt32(4);

        a.foto = (byte[])reader.GetValue(5);

        articulos.Add(a);

    }

    reader.Close();

}

}

if (dataAvailable)
{
    return new
OkObjectResult(JsonConvert.SerializeObject(articulos));
}

else
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("No hay artículos
disponibles")));
}

catch (Exception e)
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
}

```

```
    }  
}  
}
```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "Descripcion_articulo".

Dentro de la clase "Descripcion_articulo", crea una clase pública llamada "ParametroConsultaDescripcion" con la siguiente propiedad:

Descripcion: cadena de texto

Crea otra clase pública llamada "Articulo" con las siguientes propiedades:

id_articulo: entero

desc_articulo: cadena de texto

Crea otra clase pública llamada "Error" con una propiedad "Mensaje" de tipo cadena de texto.

Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Ver_articulo". Este método toma los siguientes parámetros:

Un objeto HttpRequest llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Ver_articulo".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParametroConsultaDescripcion" y almacénalo en una variable llamada "articuloDescripcion".

Extrae la propiedad "Descripcion" de "articuloDescripcion" y guárdala en una variable llamada "Descripcion".

Crea una lista de objetos "Articulo" llamada "listaDescripcion" y una variable booleana "dataAvailable" establecida en falso.

Crea una nueva conexión a SQL Server utilizando la cadena de conexión especificada.

Abre la conexión.

Crea un comando SQL para seleccionar los registros de la tabla "articulos" que coincidan con la descripción especificada.

Ejecuta el comando SQL utilizando un SqlDataReader y recorre los resultados.

Dentro del bucle, crea un objeto "Articulo", asigna los valores leídos del lector de datos y añade el objeto a la lista "listaDescripcion".

Cierra el lector de datos.

Si se encontraron datos disponibles, devuelve un resultado exitoso (OkObjectResult) que contiene la lista de descripciones de artículos convertida a JSON.

De lo contrario, devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene un mensaje indicando que el artículo no está disponible.

En el bloque "catch" fuera del cuerpo del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) con un objeto "Error" que contiene el mensaje de la excepción.

Chat GPT.

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class Descripcion_articulo
    {
        public class ParametroConsultaDescripcion
        {
            public string Descripcion { get; set; }
        }

        public class Articulo
        {
            public int id_articulo { get; set; }
            public string desc_articulo { get; set; }
        }
    }
}
```

```

public class Error
{
    public string Mensaje { get; set; }

    public Error(string mensaje)
    {
        Mensaje = mensaje;
    }
}

[FunctionName("Ver_articulo")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Ver_articulo")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParametroConsultaDescripcion articuloDescripcion =
JsonConvert.DeserializeObject<ParametroConsultaDescripcion>(requestBody);
        string Descripcion = articuloDescripcion.Descripcion;

        List<Articulo> listaDescripcion = new List<Articulo>();
        bool dataAvailable = false;

        using (SqlConnection connection = new
SqlConnection("<connection-string>"))
        {
            connection.Open();

```

```

        using (SqlCommand command = new SqlCommand("SELECT ID,
Descripcion FROM articulos", connection))
{
    SqlDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        dataAvailable = true;
        Articulo a = new Articulo();
        a.id_articulo = reader.GetInt32(0);
        a.desc_articulo = reader.GetString(1);
        listaDescripcion.Add(a);
    }

    reader.Close();
}

if (dataAvailable)
{
    return new
OkObjectResult(JsonConvert.SerializeObject(listaDescripcion));
}
else
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error("Artículo no
disponible")));
}
catch (Exception e)
{

```

```
        return new  
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));  
    }  
}  
}  
}  
}
```

Flores Castro Luis Antonio.

Crea un nuevo espacio de nombres (namespace) llamado "FunctionApp1".

Dentro del espacio de nombres, define una clase estática llamada "VerArticulo".

Dentro de la clase "VerArticulo", crea una clase pública llamada "ParemetroConsultaCantidad" con una propiedad "ID" de tipo cadena de texto.

Crea otra clase pública llamada "Articulo" con una propiedad "Cantidad" de tipo entero.

Crea otra clase pública llamada "Error" con una propiedad "mensaje" de tipo cadena de texto. Añade un constructor que tome un parámetro "mensaje".

Agrega un método público y estático llamado "Run" decorado con el atributo "FunctionName" y el nombre "Ver articulo". Este método toma los siguientes parámetros:

Un objeto `HttpRequest` llamado "req" decorado con el atributo "HttpTrigger". Configura el nivel de autorización como "Anonymous" y el método HTTP como "post". Establece la ruta como "Ver articulo".

Un objeto ILogger llamado "log".

En el cuerpo del método "Run", sigue las siguientes instrucciones:

Lee el cuerpo de la solicitud HTTP y guárdalo en una variable llamada "requestBody".

Deserializa el contenido de "requestBody" en un objeto "ParemetroConsultaCantidad" y almacénalo en una variable llamada "cantidadArticulo".

Extrae la propiedad "ID" de "cantidadArticulo" y guárdala en una variable llamada "IDArticulo".

Crea una nueva conexión a SQL Server utilizando la cadena de conexión especificada.

Abre la conexión.

Dentro de un bloque "try", sigue las siguientes instrucciones:

Crea un comando SQL para seleccionar la cantidad de artículos de la tabla "articulos" donde el ID coincida con el valor proporcionado.

Asigna el valor del parámetro "@ID" en el comando SQL.

Utiliza un SqlDataReader para ejecutar el comando y leer los resultados.

Dentro del bloque "if (reader.Read())", lee el valor de la columna "Cantidad" del lector de datos.

Devuelve un resultado exitoso (OkObjectResult) que contiene una lista vacía convertida a JSON.

En el bloque "catch" dentro del método "Run", captura cualquier excepción y devuelve un resultado de error (BadRequestObjectResult) que contiene el mensaje de la excepción.

Chat GPT.

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using System.Net;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace FunctionApp1
{
    public static class VerArticulo
    {
        public class ParametroConsultaCantidad
        {
            public string ID { get; set; }
        }

        public class Articulo
        {
            public int Cantidad { get; set; }
        }
    }
}
```

```

public class Error
{
    public string mensaje { get; set; }

    public Error(string mensaje)
    {
        this.mensaje = mensaje;
    }
}

[FunctionName("Ver_articulo")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
"Ver_articulo")] HttpRequest req,
    ILogger log)
{
    try
    {
        string requestBody = new StreamReader(req.Body).ReadToEnd();
        ParemetroConsultaCantidad cantidadArticulo =
JsonConvert.DeserializeObject<ParemetroConsultaCantidad>(requestBody);
        string IDarticulo = cantidadArticulo.ID;

        using (SqlConnection connection = new
SqlConnection("<connection-string>"))
        {

            connection.Open();

            try
            {
                using (SqlCommand command = new SqlCommand("SELECT
Cantidad FROM articulos WHERE ID=@ID", connection))

```

```
        command.Parameters.AddWithValue("@ID", IDarticulo);

    using (SqlDataReader reader =
command.ExecuteReader())
    {

        List<Articulo> listaCantidad = new
List<Articulo>();

        if (reader.Read())
        {
            int cantidad = reader.GetInt32(0);

            return new
BadRequestObjectResult(JsonConvert.SerializeObject(cantidad));
        }
    }

        return new
OkObjectResult(JsonConvert.SerializeObject(listaCantidad.ToArray()));
    }
}

catch (Exception e)
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
}
}

catch (Exception e)
{
    return new
BadRequestObjectResult(JsonConvert.SerializeObject(new Error(e.Message)));
}
}
```

}

}

Conclusiones.

A través de la realización de esta tarea pude observar que el uso de serverless es de bastante utilidad ya que se ahorran bastantes recursos y tiempo a la hora de utilizarlo. El contraste que pude identificar de manera muy significativa fue que en la tarea 7 el configurar y levantar Tomcat en la maquina virtual era algo tedioso debido a las variables de entorno, sin embargo, esta herramienta facilita muchísimo la publicación de los servicios web, yo hago la analogía de que visual studio y azure es como tener un gitlab, es decir se trabaja toda la parte de codificación en visual y en azure queda publicado. Se me hizo muy interesante la adaptabilidad de los códigos referentes al back end, en lo personal no había utilizado C# hasta la realización de esta tarea, me pareció un lenguaje hasta cierto punto sencillo y con la ayuda de chat GPT pude interpretar mejor las traducciones de código de java a C#, finalmente puedo decir que, si tomo cierto tiempo realizar esta tarea, puesto que el aprender a hacer uso de visual en conjunto con azure en ocasiones me era un poco tedioso. Aprendí bastante.